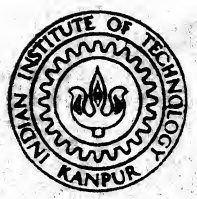
DESIGN AND IMPLEMENTATION OF GRAPHICS PACKAGE IN SIMULA - PART ONE

By

L. R. NAGAMOORTHY



DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

DECEMBER, 1984

DESIGN AND IMPLEMENTATION OF GRAPHICS PACKAGE IN SIMULA - PART ONE

A Thesis Submitted

In Partial Fulfilment of the Requirements

for the Degree of

MASTER OF TECHNOLOGY

82158

By

L. R. NAGAMOORTHY

to the

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

DECEMBER, 1984

22/12/84

C E R T I F I C A T E

CERTIFIED that the thesis entitled 'DESIGN AND IMPLEMENTATION OF GRAPHICS PACKAGE IN SIMULA - PART 1 ' has been carried out under our supervision and that it has not been submitted elsewhere for a degree.

Dr. R. Raghuram
Indian Institute of Technology
Kanpur

Dr. Sanjay G. Dhande
Indian Institute of Technology
Kanpur

Kanpur
Dec. 1984

POST GRA[illegible] OFFICE
This thesis [illegible] ed
for the awar[illegible] e of
Master of [illegible] Tech.)
in acco[illegible]
regula[illegible] an,
Insti[illegible] anpur
[illegible] 3/1/85

13 JUN 1985

CENTRAL LIBRARY
Acc. No. A 87486

EE-1984-M-NAG-DES

TO MY

PARENTS

ACKNOWLEDGEMENTS

It is with deep sense of gratitude and with immense pleasure that I thank my thesis supervisors Dr. Sanjay G. Dhande and Dr. R. Raghuram for suggesting this topic and for providing able guidance and constant encouragement without which this endeavour would not have been a success.

I shall ever be grateful to DRDO Organization for sponsoring me to the M.Tech. programme.

I would like to acknowledge the excellent co-operation provided by Dr. P.K. Bhat.

It would be a futile exercise for me to acknowledge my gratefulness to Mr. K. Siva Krishna Reddy and his family for their loving hospitality.

I would also like to appreciate excellent typing work done by Mr. C.M. Abraham.

Last, but not the least, I would like to remember the pleasant hours spent with DEC-1090.

- L.R. NAGAMOORTHY

ABSTRACT

A Graphics package GSIMULA in a high level language SIMULA has been designed and implemented on DEC 1090 system as the host machine. There are two parts in GSIMULA. The first part deals with two-dimensional graphics and general specifications. The second part deals with three-dimensional graphics and viewing transformations. The graphic display device used for this purpose is the TEKTRONIX 4006-1, a DVST type terminal. This package includes a compact command set along with usual features such as picture segmenting and picture Editing facilities. Most of the run-time errors resulting from illogical specification of parameters for graphic commands are handled interactively during the execution. A demonstration program has been developed and can be called by the user. This program illustrates typical usages of some of the graphics commands. The complete package is written in the high-level language, SIMULA and the device dependent routines are separated so that device independence can be achieved with less effort.

CONTENTS

Chapter 1

INTRODUCTION

1.1 Software of a graphics package

A graphics system is a collection of hardware and software designed to make it easier to use graphic input and output in computer programs. The design of graphics systems is a very important aspect of computer graphics. Without such systems, graphics application programs would be extremely difficult to write; only the most expert programmers would be competent to write them and their rate of software production would be very slow. It is only by constructing graphics systems that we make it possible to exploit the potential use of computer graphics.

A graphics package is a set of procedures or functions used by an application program to generate pictures on a display device and to handle graphical interaction. We have chosen to design a graphics package in a high level, machine independent language namely SIMULA. Hereafter we call this graphics package as 'GSIMULA'. GSIMULA is similar to many existing subroutine packages and it consists of a small but functionally complete set of application independent facilities for creating arbitrary views of two and three dimensional objects and for supporting interaction between an application program and its user. It is easy to use GSIMULA for specific

application areas such as data plotting, computer aided design and simulation.

The main difficulty in constructing such a package is to give adequate consideration to many issues that affect the usefulness of the package. Some of the more important of these issues are simplicity, consistency, completeness, performance and economy.

1.2 SIMULA and its features

SIMULA is an extension of the programming language ALGOL 60. The extension lies in the CLASS and CO-ROUTINE concepts and the associated reference variables, together with fully defined text handling and input/output facilities.

An important aspect of the development of modern societies is the widespread use of increasingly complex systems of men and machines. Traffic handling at airports and harbours are typical examples. It is important to be able to understand, adapt and control such existing systems, as well as to design and implement new ones. SIMULA meets the requirements of describing such systems to any desired precision. An effort has been made to provide graphics facility for this language, thus increasing its potential.

1.3 State of art of the graphics packages

Presently a number of graphics packages namely GPGS, PLOT1 GINO/F, ACM core are available. GPGS is a set of subroutines

coded in Assembly language of the host machine and once the high level language interface software module is built then GPGS is available to the user in the form of library of graphical functions, procedures. Most modern graphics packages provide some degree of device independence. Examples are OMNIGRAPH, DISSPLA and ARPA network graphical protocol.

Many researchers have proposed developing special programming languages or extending existing programming languages to support graphics. PL/I has been extended for graphics.

1.4 Objective and Scope of the present work

Our aim is to include graphics facility in the language SIMULA. The output device used is TEKTRONIX 4006-1, direct view storage tube terminal. The development of GSIMULA is the combined work of two M.Tech. projects. This is the first part.

Part one gives the details of the two-dimensional graphics system and Part two gives the three dimensional details. The facilities available in the two dimensional graphics can be divided into four groups. These are

i) Basic command set
ii) Picture segmenting facilities
iii) Picture-Editing facilities
iv) Error Handling routines.

The basic command set is explained in the next chapter. The segment is a logical unit, not necessarily contiguous either i

the display file or on the screen. It is simply a collection of display file instructions representing graphic primitives that we can manipulate as a single unit. These segments are named through integer variables which are passed as parameters to the segment control commands. Even though, generally display file segmentation is applied to refresh-displays, there are good reasons for using segmented display files with a storage tube terminal. These are :

i) it allows display regeneration without recomputation.
ii) to achieve compatibility with refresh display packages.

Picture Editing facility allows the user to give a fresh set of parameters to the command and modify the picture till the user is satisfied. The basic structure of the picture Editor consists of a loop which is executed repeatedly. After each time the picture is redrawn in the screen with the modified data and then the package processes the next graphics command, if any. The usage of Picture Editor and limitations are explained in detail in the next chapter.

Error handling is a very important aspect of any software package. Errors at the lexical and syntactic level (such as missing or extra paranthesis or parameter of the wrong data type) would normally be handled by the compiler. Other semantic errors resulting from the GSIMULA commands would be handled by GSIMULA itself during the execution thus avoiding recompilation of source program. The various messages given by this Error handling routines and the user's reactions are described in Chapter 3 alongwith the command set.

Chapter 2

SPECIFICATIONS OF GSIMULA

2.1 Block diagram of the Graphics system

Figure 2.1 shows the major components of a typical graphics system-hardware, software and data modules. The two principal hardware components are the host computer and the DPU (display processing unit). The two important data modules shown are both stored in the shared memory. The first is the DPU display program, which is written by the graphics package and is read by the DPU. The second is the application data structure which contains, among other things, a description of the objects whose images are to be displayed and is said to model the objects.

The compilation of DPU code, unlike normal compilation, takes place at the runtime of the application program.At every state, the flow of control and the parameters in the GSIMULA baged 'SOURCE' program are changed by user interactions.

2.2 Viewing operation

The graphics package includes processes which serve to transform a device independent description of an object to a device dependent display program which generates an image that is a particular view of the object.

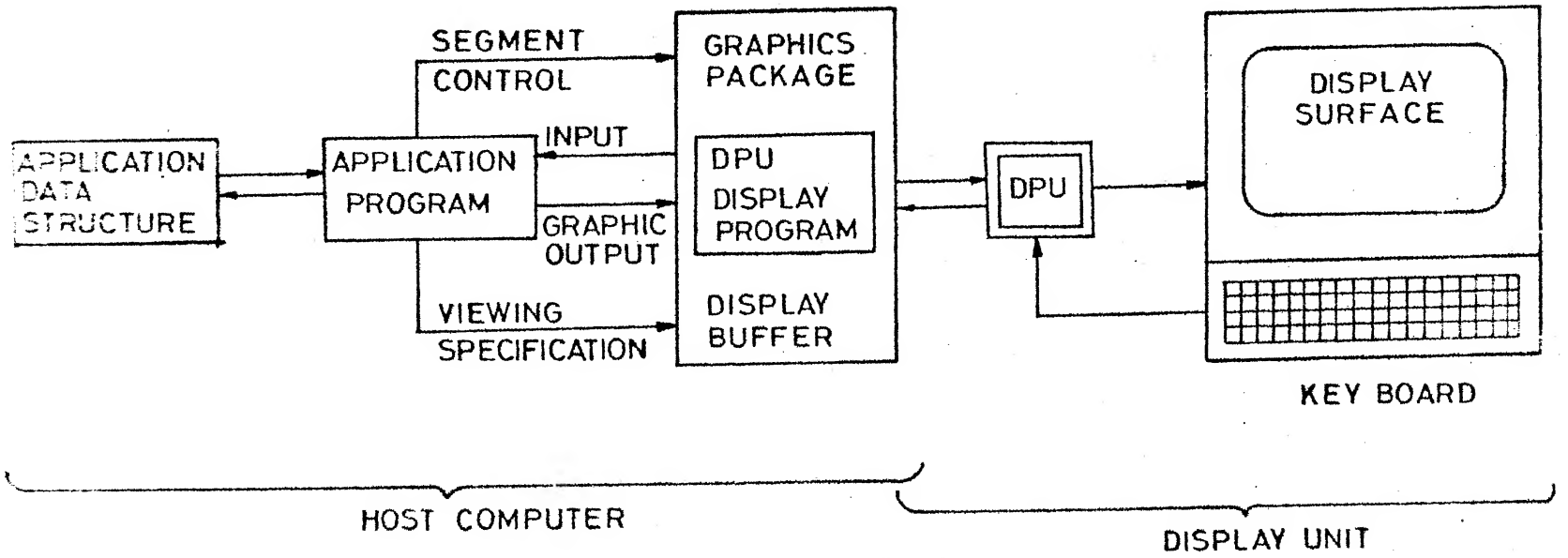


FIG. 2.1 MAJOR HARDWARE, SOFT WARE AND DATA MODULES OF A GRAPHIC SYSTEM

Figure 2.2 shows the three steps performed by these processes in detail; clipping, window to viewport mapping and generating display code in device dependent physical screen coordinates.

GSIMULA routines are oriented primarily towards vector graphics. A vector for the purpose of computer graphics is a straight line on the display surface. Each point to which a vector is drawn is defined in terms of a coordinate system.

The coordinate system exists in 'WORLD SPACE'. World space is conceptual and infinite space rather than a finite location. Figure 2.3 shows the units into which the world coordinate system is divided. These units are equal and consistent along each axis and can be thought of as any appropriate unit : Cms, Km, Degrees, Litres etc. All vectors drawn using GSIMULA are in this conceptual world space.

Four coordinates are required to define a vector. The X and Y coordinates of the starting point and ending point of the vector are required. However, one vector begins from where the previous vector ended. In that case the coordinates of the new vector are specified and the new vector is drawn from the end of the last vector to the location.

The GSIMULA routine used to draw vectors is called 'DRAWTO'. This routine has two arguments which indicate the X and Y co-ordinates of the line to be drawn from the current cursor

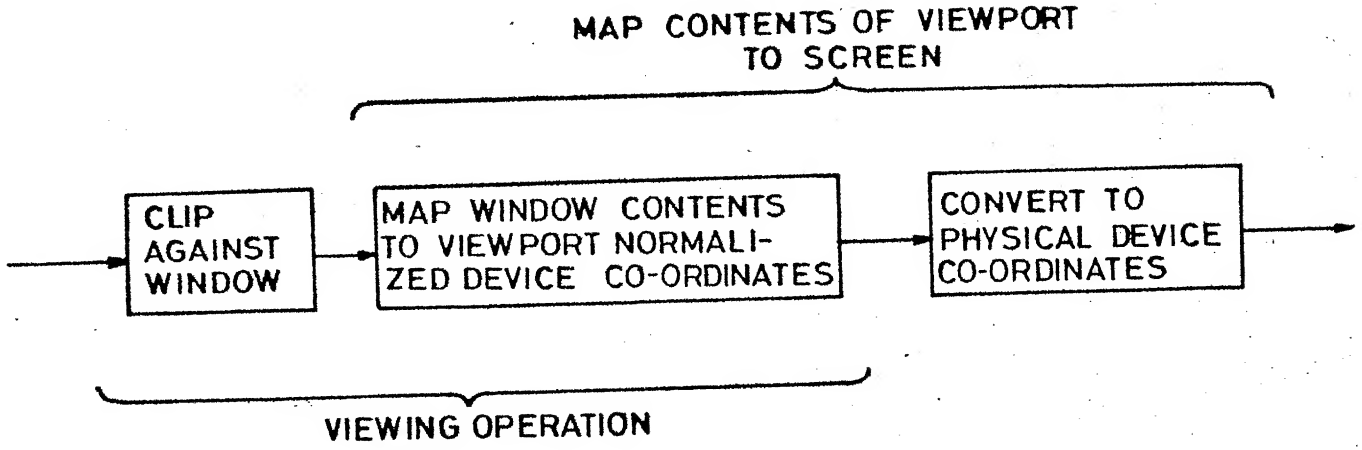


FIG. 2.2 MAPPING OBJECT'S WORLD CO-ORDINATES TO SCREEN DEVICE CO-ORDINATES

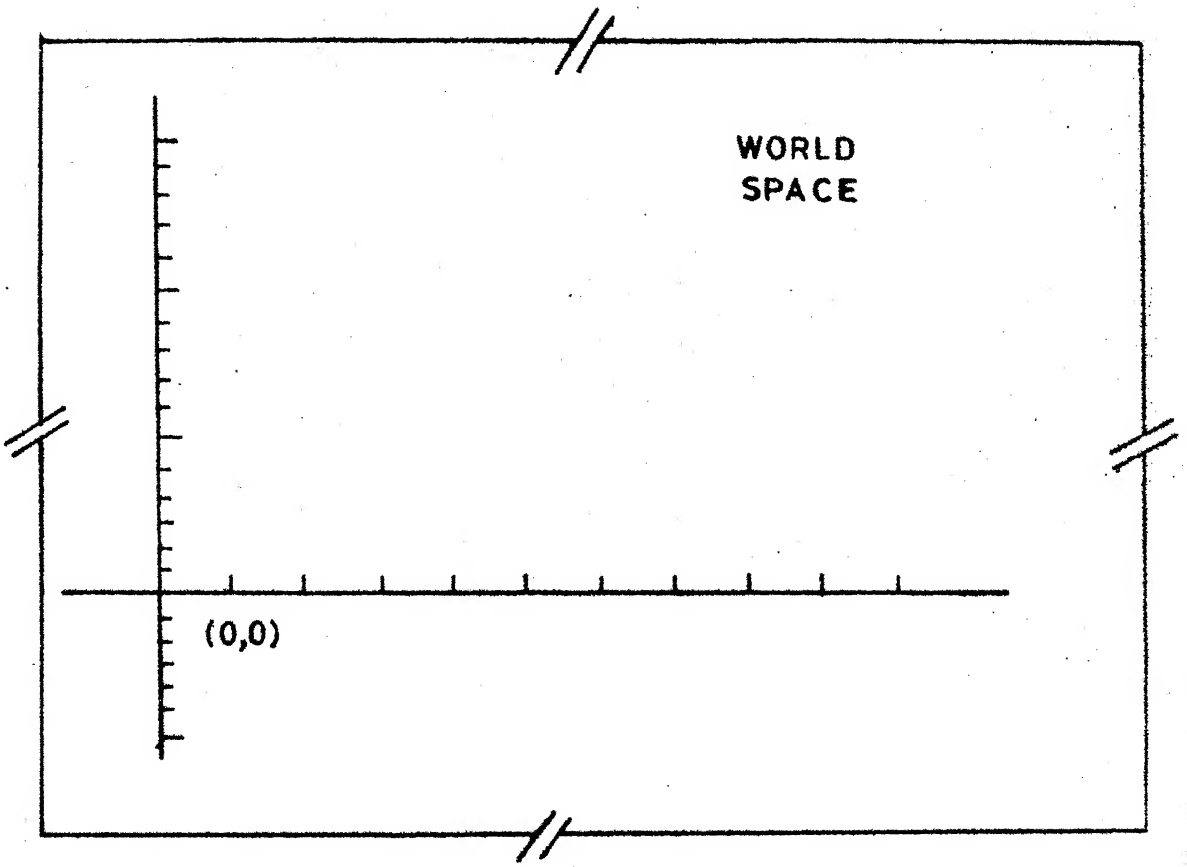


FIG. 2.3 CO-ORDINATE SYSTEM IN WORLD SPACE

position. If we donot want to draw from this current position then 'MOVETO' routine relocates the cursor without drawing a vector. Like DRAWTO, MOVETO has also two arguments which specify X and Y coordinates of the point to be moved to.

2.2.1 Window/viewport transform

The portion of the world space to be viewed is called the 'WINDOW'. The window is a rectangular subspace of world space or the specific portion of the coordinate system to be viewed. The corners of window are defined in world coordinate system.

To be displayed, a window must be projected to a display surface. A 'VIEWPORT' is the specified area on the display surface to which a window is projected. The default size of the viewport is the full display surface of the output device. The process of projecting vectors drawn in a portion of world space on the display surface is called the WINDOW/VIEWPORT transform.

2.3 How to use GSIMULA?

A Graphics program which uses GSIMULA routines is handled like any other program. That is the steps necessary to prepare a graphics program for execution on a host computer are the following :

i) Establish communication with the host

ii) Create a source program with prefixing CLASS GSIMULA

iii) Compile the program

iv) Link it with external libraries

v) Set tty No CRLF, WIDTH ; 80

vi) Execute .

The source program consists mainly of calls to GSIMULA procedures which perform graphic functions. The source language is used to control the program loops and to perform the mathematical functions necessary for graphics.

2.4 Data structures

The primary data structures used in this package are CLASS concept and arrays, since it is possible to define the size of the array dynamically and random accessing arrays are preferred against possible data structures like doubly linked lists which require more memory and serial accessing.

Chapter 3

IMPLEMENTATION DETAILS

In previous chapters, we have taken our first look at interactive graphics application, programming, hardware and software. We have focussed on the use and design of a simple subroutine based graphics package. This package is callable from within standard high level languages in order to make application programs transportable among all computer systems supporting the package. In this chapter, we shall see how the package is implemented.

3.1 Primitives

In GSIMULA, MOVETO (PX,PY) and DRAWTO (PX,PY) are the two primitive commands. All other commands are developed or constructed using these two primitive graphics outputs.

Transformations

Each set of coordinates passed from the output function is processed by GSIMULA before being passed to the device driver for display building. All coordinate inputs are processed in four steps as below :

i) Transformed by the system transformation matrix.

ii) Clipped to the current users window if clipping is enabled.

iii) Mapped to viewport coordinates within the currently specified viewport.

iv) The transformed coordinates (V') are obtained by post multiplying the input coordinates (V) by the system transformation matrix.

For the MOVETO Command, the above four steps are carried out and the latest values of X and Y coordinates are stored and no other parameters are passed to the DPU of the display device. For the DRAWTO Command, after getting the clipped coordinates, the following steps are executed.

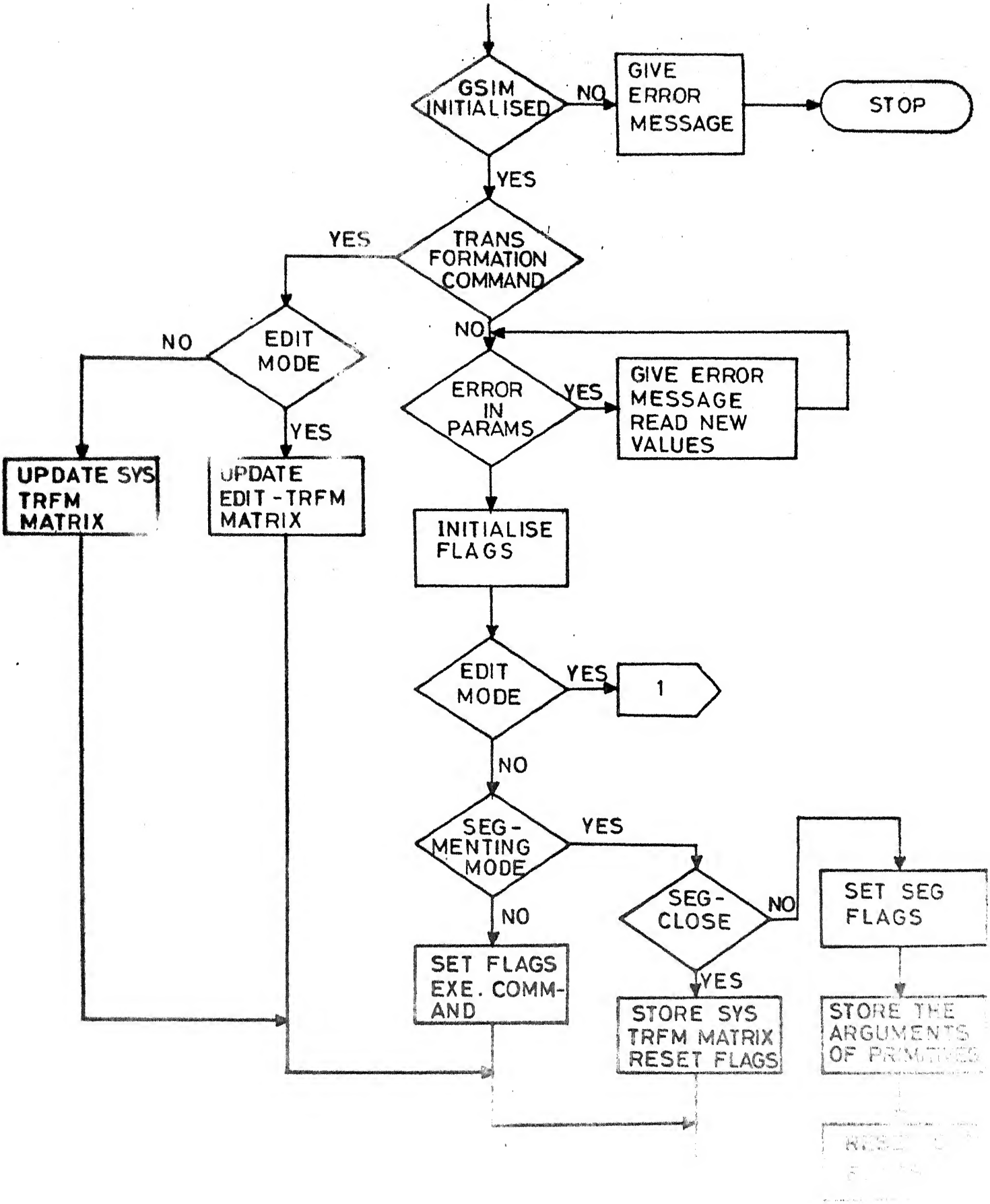
i) Enter graphics mode

ii) Calculate the parameters to be sent to the display device according to the specified format

iii) Pass those parameters into register 3 of the host computer

iv) Leave the graphics mode.

Since direct I/O handling facility is available in SIMULA all the above steps are carried out in the high level language otherwise MACRO routines are to be used for the above purpose. Whenever the transformation procedures are invoked (for example ROTATEO, SCALE etc), the system transformation matrix is modified appropriately.


GSIM INITIALISED
NO
GIVE ERROR MESSAGE
STOP
YES
TRANS FORMATION COMMAND
YES
EDIT MODE
NO
YES
NO
UPDATE SYS TRFM MATRIX
UPDATE EDIT-TRFM MATRIX
ERROR IN PARAMS
YES
GIVE ERROR MESSAGE READ NEW VALUES
INITIALISE FLAGS
EDIT MODE
YES
1
NO
SEG-MENTING MODE
YES
NO
SET FLAGS EXE. COMM-AND
SEG-CLOSE
NO
YES
STORE SYS TRFM MATRIX RESET FLAGS
SET SEG FLAGS
STORE THE ARGUMENTS OF PRIMITIVES

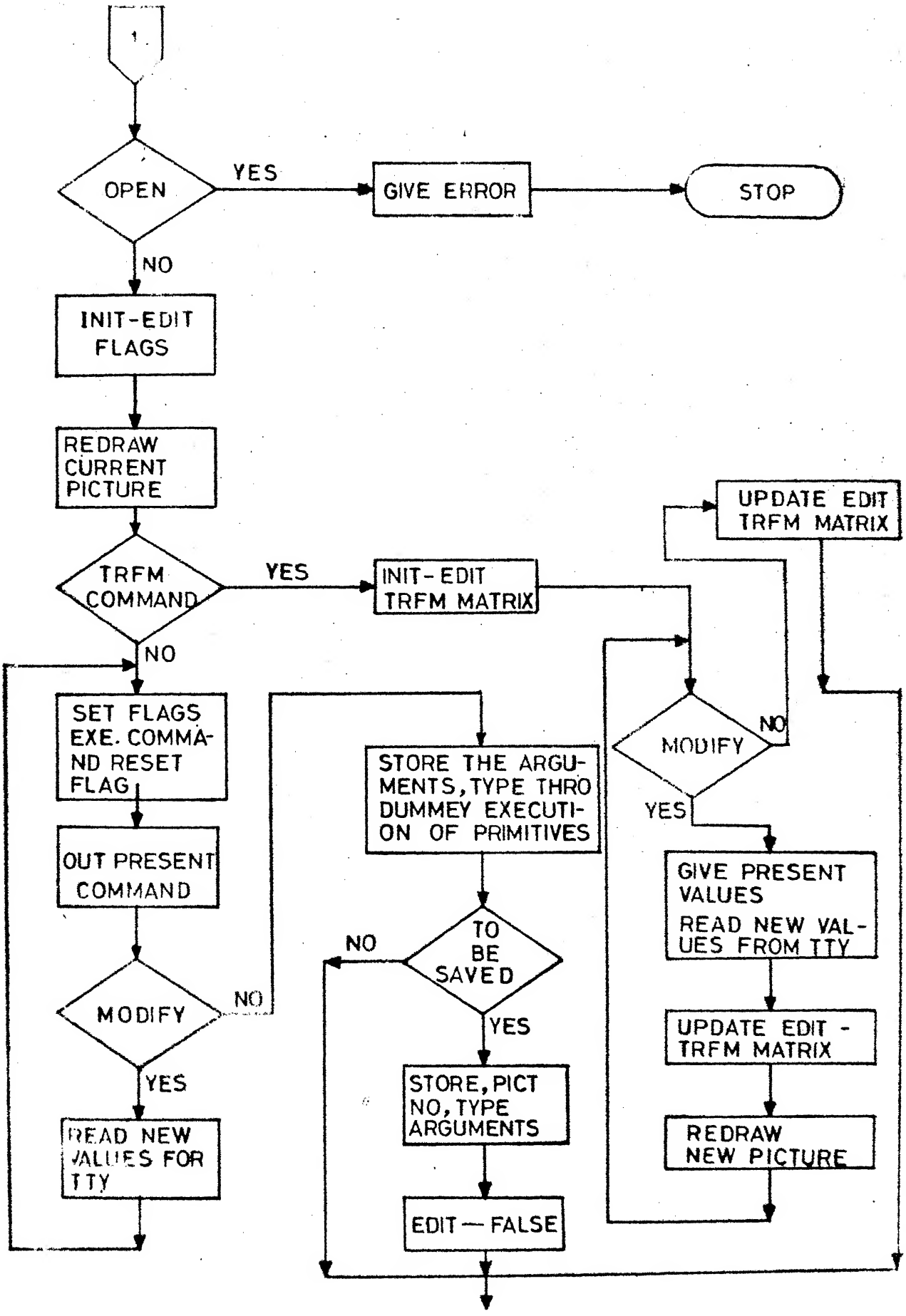


FIG. 3.2 GSIMULA COMMAND PROCESSED IN EDIT MODE

For example, when ROTATE O(Q) is called, the present transformation matrix S is multiplied with the matrix T where T is represented by

$$T = \begin{bmatrix} \cos Q & \sin Q & 0 \\ -\sin Q & \cos Q & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

so that the new system transformation matrix is given by

$$[S]_{new} = [S]_{old}^{*} [T] .$$

3.2 Implementation of picture segmenting

Whenever a segment is opened, a flag OPEN is set to true. So when a procedure is executed, for example CIRCLE (PX,PY,R), the arguments of the primitives which constitute this procedure are stored in a array. The device driver routines which output the picture on the screen are not invoked. When the transformation commands are executed under segmenting, the system transformation matrix is updated for that particular segment and so also stored when the segment is closed. When the segment is posted by calling the command POST(N), the primitive commands are interpreted, multiplied by the corresponding transformation matrix, and then passed on to the device driver. The list of GSIMULA commands which cannot be used for segmenting should be noted by the user.

3.3 Picture Editing

This is slightly complex and an interesting feature of the GSIMULA package. It is possible to develop a picture on line by an experienced programmer under the picture editing command. The first command to be called for picture Editing is EDIT-PICT(N); N is the picture number. So, in a program different pictures can be edited under different picture numbers. Now, all the GSIMULA commands followed by this and upto the GSIMULA command END-PICT(N) or END-SAVE-PICT(N) will be repeatedly executed giving their present arguments on the display surface, wait for the new sets of arguments if the user has replied 'Y' for modification.

Implementation

As soon as the Edit-PICT(N) is called, a boolean EDIT is set to True. Then, any GSIMULA command is processed as per the flow chart given in Fig. 3.2. When the edited picture is to be redrawn (if saved), then the edited picture transformation matrix is multiplied by the present system transformation matrix, then interpreted accordingly.

3.4 Error handling

To minimise the number of recompilation of source program due to semantically incorrect parameters passed to GSIMULA commands, on line error handling scheme has been adopted. For each GSIMULA command, the validity for the arguments, for

example radius of the circle specified as negative or zero, is verified and then proceeded for execution. If there is an error in the parameters passed, the program gives the command name, present values of parameters and waits for the new set of data. So, this scheme reduces the GSIMULA errors to less than six, which is convenient to the user. After the user gives new values, program continues execution with fresh values.

The commands that can be called by the GSIMULA user are given below.

3.5 Description of the command set

1. ARC(P, STANGLE,ENDANGLE)

Purpose

Draws an arc from the current cursor position with starting angle STANGLE to the ENDANGLE with radius P.

INPUTS

P The radius of the arc (real)

STANGLE starting angle of the arc (integer)

ENDANGLE ending angle of the arc (integer)

Errors

If an error such as $P \leq 0$ occurs the package will put on out a message along with a request to give correct value of P. The user is expected to enter interactively the acceptable value of P. This, in short, is now referred to as the interactive handling of errors.

2. ARC3PT (P,Q,R,S,T,U)

Purpose

Draws an arc passing through the three points (P,Q), (R,S) and (T,U) starting from (P,Q) in anticlockwise direction.

INPUTS

P,Q	first point coordinates (real)
R,S	second point coordinates (real)
T,U	third point coordinates (real)

Errors

None.

3. ARCPANEL (P,STANGLE,PENDANGLE,NL,SLOPE)

Purpose

Fills up the arc area enclosed by present cursor position, STANGLE,PENDANGLE with radius P. The number of lines are equal to NL and slope is the angle with respect to X axis (in degrees)

INPUTS

P,STANGLE,PENDANGLE	as per ARC command
NL	No. of lines (integer)
SLOPE	slope of the line (integer) in degrees

Errors

Some of the errors are interactively handled.

4. BELL

Purpose

Causes the device's bell to sound.

5. BREAK

Purpose

Stops the execution of the program and continues after the user hits the carriage return.

6. BRINGCHAR (X,Y,C)

Purpose

Draws the character specified by 'C' in the world coordinate (X,Y). Scaling of the character depends on current setting of SCALECHAR command.

INPUTS

X,Y	The X and Y coordinates of the point where the software generated character 'C' is to be drawn (real)
C	The character which is to be drawn. (C is a character data type).

Errors

None.

7. BRINGDIGIT (X,Y,I)

Purpose

Draws the integer specified by I in the world coordinate (X,Y). Scaling of the integer depends on current setting of SCALEDIGIT command.

INPUTS

X,Y	The x and y coordinates of the point where the software generated integer denoted by I is to be drawn (real)
I	integer to be drawn (integer)

Errors

None.

8. BRINGTEXT (X,Y,T)

Purpose

The text or character string denoted by T is written with the first character starting at the world coordinate (X,Y). The size of the character is determined by SCALECHAR command.

Inputs

X,Y	coordinates (real)
T	character string (text)

Errors

None.

9. CHDRAWTO (x,y,I)

Purpose

Draws chained line from the current cursor position to the point (x,y). 'I' indicates the desired pattern.

INPUTS

x,y	Coordinates of point upto which the chained line is to be drawn. (real)
I	integer (0 through 5) indicating the desired pattern (integer)

Errors

I $\geq$ 6, solid line is drawn.

10. CIR3PT (P,Q,R,S,T,U)

Purpose

Draws the circle which passes through three points (P,Q), (R,S) and (T,U).

Inputs

P,Q	Coordinates of the first point (real)
R,S	Coordinates of the second point (real)
T,U	Coordinates of the third point (real)

Errors

None.

11. CIRCLE (PX,PY,PR)

Purpose

Draws the circle with (PX,PY) as centre and PR as radius.

Inputs

PX,PY	Coordinate of the centre of the circle(real)
PR	Radius of the circle (real)

Errors

Some errors interactively handled.

12. CIRPANEL (PX,PY,PP,PNL,PANGLE)

Purpose

Fills up the circular area (centre being PX,PY and radius PP) with lines whose angle is equal to PANGLE.

PNL is the number of lines filling up the circular area.

Inputs

PX,PY,PP	as per CIRCLE specification
PNL	number of lines (integer)
PANGLE	slope of the lines (integer)

Errors

Run time errors interactively handled.

13. CLOSE SEGMENT(N)

Purpose

Closes the segment (N) which is already opened.

Input

N	denotes the segment number (integer).

Error

If segment (N) is not already opened,

Error is detected and execution is stopped.

14. CLIPSTATUS (N)

Purpose

Those parts of the picture lying outside the window may be made visible if desired by the user. This is termed clipping to the window. Clipping may be selectively enabled or disabled by this call.

Input

N when N = 0, clipping is disabled otherwise enabled

Errors None.

15. DRAWCHAR(X,Y,C)

Purpose

Draws the character (generated by hardware) denoted by C at the point (x,y)

Inputs

X,Y Coordinates of the point where the character C is to be written (real)

C Character which is to be written (character)

Errors

None.

16. DRAWTEXT(S)

Purpose

Writes the text string (hardware generated) denoted by S from the present cursor position.

Input

S character string (Text)

Errors

None.

17. DPOLAR(PR,PANGLE)

Draws an arc from the current cursor location to the specified polar coordinates.

Input

PR Distance from the origin specified in world coordinates (real)

PANGLE Angle (degrees) component of point to which arc is drawn (integer).

Errors

Run time errors handled interactively.

18. DRAWTO(X,Y)

Purpose

Draws a vector from the current cursor location to a specified point

Inputs

X x-coordinate of point to which vector is drawn (real)

Y y-coordinate of point to which vector is drawn (real)

Errors

None.

19. EDIT-PICT(N)

Purpose

After execution of this command, all the graphics command following this are repeatedly executed till the user is satisfied. This is dealt in detail in the next chapter.

Input

N an integer denoting the picture number for editing (integer)

Error

Run time errors causes the program to stop.

20. END-Edited (N)

Purpose of this command is explained in the next chapter.

21. EXIT

Purpose

The transformation matrix is restored to identity matrix.

22. END-SAVE-EDITED(N)

Explained in the next chapter.

23. GRAPHPLOT (N,Vx,Vy,Sx,Sy)

Purpose

User can call this to draw a graph

Inputs

N	No. of points to be plotted (integer)
Vx	Real array containing the x-coordinate of the points
Vy	Real array containing the y-coordinate of the points
Sx	Character string to be written in x-axis
Sy	Character string to be written in y-axis

Errors

None.

24. INIT-BUFF(N)

Purpose

Display buffer to be allocated by the user when he is employing picture segmenting.

25. INIT-GSIM

Purpose

First GSIMULA Command to be invoked by the user.

26. MOVETO(PX,PY)

Purpose

Moves the cursor to a specified point without drawing a vector

Inputs

X x coordinate of point to which cursor is to be moved

Y y coordinate of point to which cursor is to be moved

Errors

None.

27. MPOLAR(R,ANGLE)

Purpose

Moves the cursor to the specified polar coordinates

Inputs

R The distance from the origin

ANGLE Angle component of the point where cursor moves

Errors

Run time errors are handled interactively.

28. NEWPAGE

Purpose

Provides a clean surface for display of output

Errors

None.

29. OPEN-SEMENT(N)

Purpose

Indicates that segment (N) is being opened.

Input

N integer denoting the segment No.

Error

IF already a segment is open, error detected and execution is stopped.

30. POLY(N,Vx,Vy)

Purpose

Draws a polygon

Inputs

N Size of Vx array and Vy array

Vx x coordinate of points to be drawn (real)

Vy y coordinates of points to be drawn (real)

Error

Run time errors handled interactively.

31. POLYPANEL(N,Vx,Vy,NL,SANGLE)

Purpose

Fills the polygon with straight lines whose slope is equal to SANGLE . The number of lines are equal to NL.

Inputs

N Size of Vx array and Vy array (integer)

Vx x-coordinates of points to be drawn (real)

Vy y-coordinates of points to be drawn (real)

NL Number of lines (integer)

SANGLE Slope of the line (integer)

Error

Run time errors are handled interactively.

32. RDRAWTO (Px,Py)

Purpose

Draws a vector from the current cursor position to a specified point, the displacement in x and y direction being relative.

Inputs

Px	Amount by which in X coordinate is to be drawn (real)
Py	Amount by which in Y coordinate is to be drawn (real)

Error

None.

33. RECTANGLE (Px,Py,P,Q)

Purpose

Draws a rectangle at point (Px,Py) with sides P and Q

Input

Px	x-coordinate of point where the rectangle is to be drawn (real)
Py	y-coordinate of point where the rectangle is to be drawn (real)
P	Length of the rectangle (real)
Q	Height of the rectangle (real)

Error

Run time errors interactively handled.

34. REFLECT-X

Purpose

Reflects the picture with respect to X-axis.

35. REFLECT-Y

Purpose

Reflects the picture with respect to Y-axis.

36. REFLECT-XOY

Purpose

Reflects the picture both in X and Y axis.

37. REGPOL (Px,Py,PRAD,PNUMSID)

Purpose

Draws a regular polygon of number of sides equal to NUMSID with centre (Px,Py) in the world coordinate. PRAD is the radius (distance from the centre to the maximum distant point on the circumference).

Inputs

PX	x-coordinate of the centre (real)
PY	y-coordinate of the centre (real)
PRAD	radius (real)
PNUMSID	No. of sides (integer)

Error

Run time errors interactively handled.

Acc. No. A 87486

38. REGPOLPANEL (PX,PY,PRAD,PNUMSID,PNL,PANGLE)

Purpose

Fills up the regular polygon area with straight lines whose slope is PANGLE

Inputs

Px,Py,PRAD,PNUMSID	specified as per REGPOL
PNL	No. of lines (integer)
PANGLE	Slope (integer)

Error

Run time errors interactively handled.

39. REPLAY-EDITED (N)

Purpose

Redraws the edited picture No 'N' if it was saved

Error

Illegal use of this command stops the program execution.

40. RMOVETO (PX,PY)

Purpose

Moves the cursor relatively with PX units in X direction and PY units in y-direction, relatively.

Inputs

PX	Distance that is to be moved in x-direction (real)
PY	Distance that is to be moved in y-direction (real)

Errors

None.

41. ROTATEO (Q)

Purpose

Indicates the rotation to be applied with respect to origin

Input

Q	Extent of rotation in degrees (integer).

42. ROTATEPT (Q,Px,Py)

Purpose

Indicates rotation to be applied with respect to the point.

Input

Q	Extent of rotation in degrees (integer)
Px	x-coordinate of point with respect rotation is applied (real)
Py	y-coordinate of point with respect rotation is applied (real)

Errors

None.

43. SCALE (Px,Py)

Purpose

Specifies a scale factor applied to the coordinate system

Inputs

Px	Scale factor applied to X axis (real)
Py	Scale factor applied to Y axis (real)

Errors

None.

44. SCALEPT (Sx,Sy,Px,Py)

Purpose

Specifies a scale factor applied to the coordinate system with respect to the point (Px,Py)

Inputs

SX	Scale factor in x-axis (real)
SY	Scale factor in y-axis (real)
Px	x-coordinate of point with which scaling is applied (real)
Py	y-coordinate of point with which scaling is applied (real)

Errors

None.

45. SCALE-UNI (Px)

Purpose

Specifies a scale factor applied to both x and y axes

Input

Px	Scale factor (real)

Error

None.

46. SHEAR-X (P)

Purpose

Specifies the shearing factor applied to x axis.

Input

P	Extent to which the shearing is applied (real)

Errors

None.

47. SHEAR-Y (P)

Purpose

Specifies the shearing factor applied to y axis

Input

P Extent to which shearing is applied (real)

Errors

None.

48. SETWINDOW (xLOW,XHIGH,yLOW,yHIGH)

Purpose

Specifies the portion of the coordinate system to be viewed

Inputs

XLow Minimum x coordinate of window (real)

XHigh Maximum x coordinate of window (real)

YLow Minimum y coordinate of window (real)

YHigh Maximum y coordinate of window (real)

Errors

Run time errors handled interactively.

49. SETVIEWPORT (XLow,XHigh,YLow,YHigh)

Purpose

Defines location of output on display surface

Inputs

XLow Minimum x coordinate of viewport (real)

XHigh Maximum x coordinate of viewport (real)

YLow Minimum y coordinate of viewport (real)

YHigh Maximum y coordinate of viewport (real)

Error

Run time errors handled interactively.

50. SQUARE (Px,Py,PR)

Purpose

Draws a square at the point (Px,Py) of side PR

Inputs

Px	x-coordinate of the bottom left corner of the square (real)
Py	y-coordinate of the bottom left corner of the square (real)
PR	Side of the square (real)

Error

Logical error in specification of the parameter handled interactively.

51. TRANSLATE (Px,Py)

Specifies the extent of translation in x and y coordinate axes

Input

Px	Amount of translation in x-axis (real)
Py	Amount of translation in y-axis (real)

Error

None.

52. SCALECHAR (Sx,Sy)

Purpose

Establishes the size of the software generated characters

Inputs

Sx	Specifies the scale factor in x axis
Sy	Specifies the scale factor in y axis

Error

None.

53. SCALEDIGIT (Sx,Sy)

Purpose

Establishes the size of the software generated digits

Inputs

Sx	Specifies the scale factor in x axis
Sy	Specifies the scale factor in y axis

Error

None.

Chapter 4

ILLUSTRATIVE EXAMPLES

A demonstration program has been developed in which the typical usage of GSIMULA Commands are shown. This program can be called by the user by name 'DEMO'.

Plate 1 : Here, the heading GSIMULA indicates the use of the Command BRINGTEXT; the size of the character is 0.5 to normal size. WELCOME is generated by successively changing the scale factor in y-axis alone and placing them with a fixed distance apart in the x-axis. The first letter W is the normal size of the character.

Plate 2 : This shows the capability of interactively handling errors. SETWINDOW (+100, -100, 100, 1000); In this command xL is greater than xh. During the runtime, this error is detected and the message is given on the terminal. After the user supplies the correct set of parameters, execution continues.

In the next statement, the side of the rectangle is given in Negative. This error is detected and after getting correct set of values, Execution continues.

Plate 3 : This shows a pattern produced by placing 30 points equally spaced on the circumference of a circle and then joins each point to every other.

Plate 4 : This shows the usage of viewport facility, Rotate with respect to origin, and uniform scaling. A square is successively scaled down, rotated thro' constant angle and placed in a part of the display surface using SETVIEWPORT command.

Plate 5 : This diagram is called as 'PIN and COTTON PICTURE'. This is obtained by calculating the coordinates of N equidistant points around the edges of a square. Then pairs of points are joined, according to the following rule.

P(I) P(I) = 1,2, ..., N is joined to P(J) for all I and J such that (J-I) is a FIBONNACCI Number less than 4N, substraction being carried Modulo 4N.

Plate 6 : This shows the panelling facility. The Display surface is divided into four Quadrants. In the first Quadrant, a non convex polygon is drawn by the Command POLY (N, Vx, Vy). Then this area is filled up by 40 straight lines whose slope is 45°. In the second, a Regular polygon and filling up that area are shown. In the third, circular area is filled up with 20 straight lines with a slope of 90°. And the final, circular area is filled up with 40 straight lines whose slope is 0 degree.

Plate 7 : This shows the effect of shear in x and ý directions, uniform scaling, reflections in x, y and xy axes.

Plate 8 : This is with respect to the use of RDraw to command. The user is requested to give the arguments of RDrawto. After getting the parameters, it draws that particular line and waits for next set of data for'RDrawto'. When the data supplied by the user exceeds 1000, then this procedure is skipped and goes to the next plate.

Plate 9 : This shows, how the ROTATEPT, SCALEPT Commands can be used. Here, a square is drawn at (50,50). This is rotated with respect to the point (75,50) through an angle 30 degrees. Then scaling in y direction (1.2) is applied and then it is rotated with respect to the point (50,50) through an angle -30 degrees.

Plate 10: This shows the facility of different patterns of chain lines available in GSIMULA. They are numbered 0 through 5. The

Plate 11: This shows the facility of panelling the area of an ARC.

Chapter 5

CONCLUSION

5.1 Technical summary

GSIMULA, a subroutine based graphics package has been designed, written in SIMULA and has been implemented on the DEC 1090 system using TEKTRONIX 4006-1, a DVST terminal. This package has two parts. First part deals with two dimensional and the second with three dimensional viewing. This package includes a compact but functionally complete command set along with picture segmenting, Picture Editing and a powerful error handling mechanisms. All the commands have been tested for typical parameters. The algorithms used are simple and involve simple coordinate geometry. The complete source code is in SIMULA. The size of the program is one hundred and seventy blocks.

5.2 Further scope

Picture segment libraries can be included. A software interface module can be developed so that the graphics library can be used by other high level languages such as PASCAL, FORTRAN and PL/I. GSIMULA does not deal with application dependent modelling or higher level utilities such as graph plotting packages. Response surface and other statistical results obtained in simulation can be displayed in graphics mode rather than in tabular form. This can be done using GSIMULA.

The limitations of the subroutine based graphics package include the clumsy 'parameter list', syntax and the lack of compile time checking. It would be far preferable to extend a compiled language with graphic data types and operators. The package also can be extended to raster graphic terminals by modifying the primitive commands.

APPENDIX

The following points are to be noted before using GSIMULA.

1. The default size of the viewport is the full display surface of the terminal. The default window is the same size as the default viewport. (A 'default' is a condition automatically set by GSIMULA, when it is initialized. A default condition need not be specified except to restore the default after it has been altered).

2. Picture Segmenting and Picture Editing functions should not overlap, i.e., when a Picture Segment is open, Picture Editing cannot be used and vice-versa.

3. When more than one segment is used, different Segment Number should be allotted for each segment. (can be any integer, positive or negative). Similarly for Picture Editing also.

4. The list of Errors which cannot be handled on line are given below. Once these errors occur, the source program should be corrected accordingly, recompiled and then be executed.

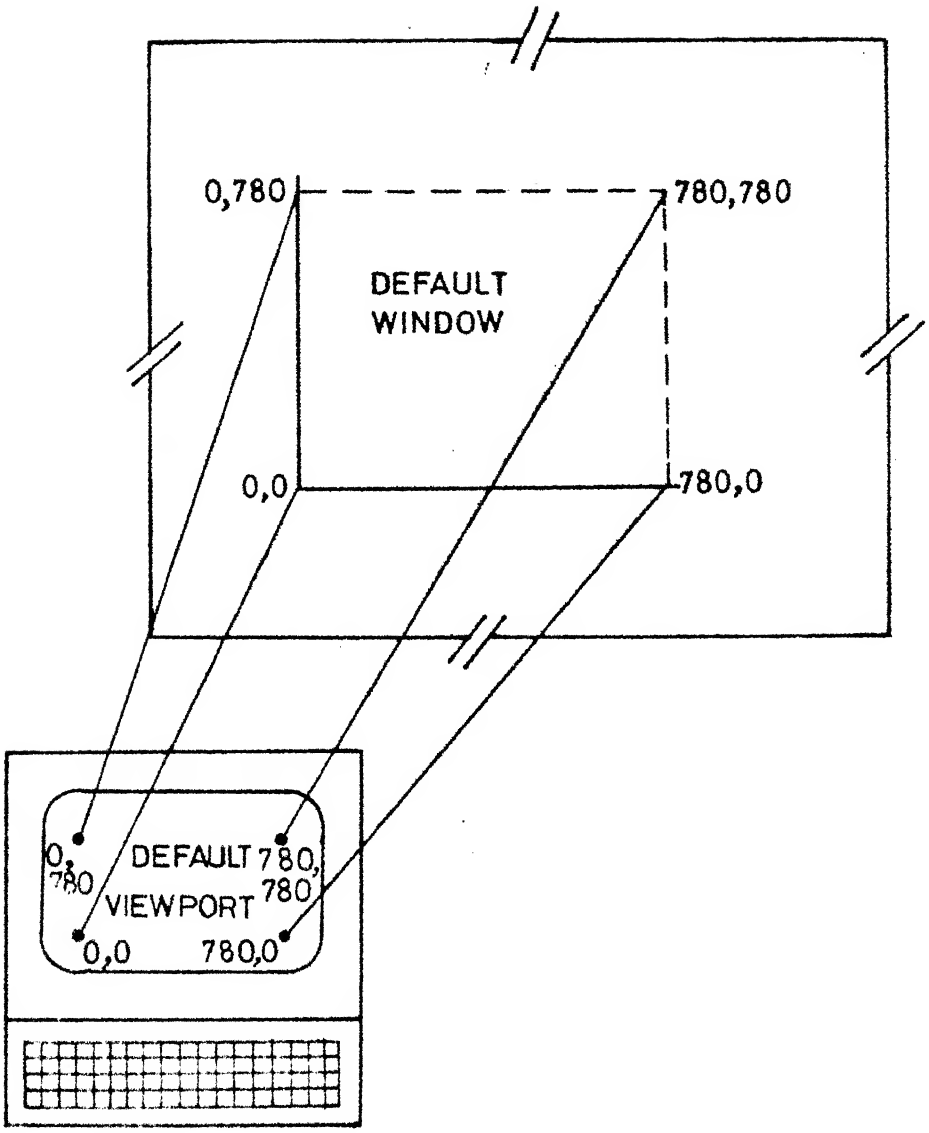


FIG. A.1 PROJECTION OF WINDOW ONTO VIEWPORT

a) GSIM ERROR No.1 : This indicates that user tried to open a segment without closing the previous segment.

b) GSIM ERROR No.2 : This indicates that user tried to edit a new picture without Ending the previous picture already in Edit mode.

c) GSIM ERROR No.3 : Indicates that user tried to overlap segmenting and Picture editing.

d) GSIM ERROR No.4 : Indicates that the user tried to close a Picture Segment which is not existing.

e) GSIM ERROR No.5 : Indicates that the user tried to end a nonexisting edited picture.

5. The primitive command DRAWTO (X2,Y2) is given below.

During the execution of MOVETO (X1,Y1) command, these world coordinates are converted into screen coordinates through WINDOW to VIEWPORT transformations and new x1,y1 are obtained. For the DRAWTO (X2,Y2) command the following codes in GSIMULA are executed.

```
Procedure  DRAWTO (X2,Y2); Real x2,y2
Begin
   OUTCHAR (CHAR(29));
   TRANSMIT-COORDINATES(x1,y1);
   TRANSMIT-COORDINATES(x2,y2);
   Outchar(char(31));
end;
```

```
Procedure  TRANSMIT-COORDINATES(x,y);
Real x,y ;
Begin
   integer xl,yl;
   xl: = x//l; yl: = y//l;
   OUTCHAR(CHAR(yl//32+32));
   OUTCHAR(CHAR(MOD(yl,32)+96));
   OUTCHAR(CHAR(xl//32+32));
   OUTCHAR(CHAR(MOD(xl,32)+64));
   Breakout image;
End;
```

REFERENCES

1. BIRTWISTLE, G.M., DAHL, O.J., MYHRHAUG, B., NYGAARD, K., 'SIMULA BEGIN'.
2. STEVEN HARRINGTON, 'COMPUTER GRAPHICS A PROGRAMMING APPROACH'.
3. DEC SYSTEM 10 ASSEMBLY LANGUAGE HANDBOOK.
4. FOLEY, J.D., VAN DAM, A., 'FUNDAMENTALS OF INTERACTIVE COMPUTER GRAPHICS'.
5. DEC SYSTEM 10 SIMULA LANGUAGE HANDBOOK.
6. PLOT 10 INTERACTIVE GRAPHICS LIBRARY.
7. MICHENER, C., FOLEY, D., 'SOME MAJOR ISSUES IN THE DESIGN OF THE CORE GRAPHICS SYSTEM', ACM COMPUTING SURVEYS, VOLUME 10, NUMBER 4, DECEMBER 1978.
8. NEWMAN, M., SPROULL, F., 'PRINCIPLES OF INTERACTIVE COMPUTER GRAPHICS'.
9. MUDUR, S.P., 'GENERAL PURPOSE GRAPHICS SYSTEM USER'S MANUAL.
10. TEKTRONIX 4006-1 COMPUTER DISPLAY TERMINAL. SERVICE INSTRUCTION MANUAL.

PROGRAM LISTING.

: The Program listing is attached with the following copies:

1) Copy given to supervisor, Dr.Sanjay G.Dhande.
2) copy given to P.G. Section.
3) Copy retained by the author,L.R.Nagamoorthy.

```
*********************************************
*    This program was developed  as a       *
*    part of M.Tech THESIS.                 *
*                                           *
*    Title:'Design And Implementation of    *
*    Graphics Package in SIMULA - Part ONE.*
*                                           *
*    VERSION 0.0.                           *
*                                           *
*    AUTHOR :  L.R.Nagamoorthy.             *
*                                           *
*    Date   : Dec 22 , 1984.                *
*********************************************
```

```
OPTIONS(/E);
EXTERNAL PROCEDURE rubout;
!THIS PROGRAM HAS BEEN DEVELOPED AS PART OF M.TECH THESIS
TITLE OF THESIS IS "DESIGN AND IMPLEMENTATION GRAPHICS PACKAGE
IN SIMULA-PART ONE".
VERSION 0.0   AUTHOR L.R.NAGAMOORTHY .DEC ,1984 ;
CLASS gsimul;
BEGIN
    CLASS st_atus;
    BEGIN
        BOOLEAN ro_tateo,ro_tatez,sh_ear ,tr_late;
    END;
    CLASS wi_ndow;
    BEGIN
        REAL wxl,wxr,wyb,wyt;
    END**XLOW XHIGH,YLOW,YHIGH OF WINDOW ARE STORED**;

    CLASS vi_ewport;
    BEGIN
        INTEGER vxl,vxr,vyb,vyt;
    END**XLOW,XHIGH,YLOW,YHIGH OF VIEWPORT ARE STORED HERE**;

    CLASS wv_constant;BEGIN REAL wvxm,wvxa,wvym,wvya;END;
    CLASS cl_ipconstant;BEGIN REAL clipxl,clipxr,clipyb,clipyt;END;
    CLASS ol_ddvst;BEGIN REAL cx,cy;END;
    CLASS ol_dpolar;BEGIN REAL  rlast,alast;END;
    CLASS pr_evious;BEGIN  REAL tx,ty;END;
    CLASS e_d_status;
    BEGIN
        CHARACTER ch;BOOLEAN satisfied;INTEGER count;
    END**FOR EDIT MODE CH USED FOR READING CHAR.FROM TTY,
    BOOLEAN IS A FLAG
    INDICATING
    USER IS SATISFIED WITH THE LATEST PARAMETER,
    COUNT KEEPS TRACK FOR STORING THE EDITED VALUES**;

    CLASS t_tyin;BEGIN REAL r1,r2,r3,r4,r5,r6;
        INTEGER i1,i2,i3,i4,i5,i6;
    END**IN EDITMODE/ERRORS THESE VARIABLES ARE
    USED TO STORE THE VALUES FROM THE TTY**;
    BOOLEAN Open,master,clip_flag,sqpat_left;
    BOOLEAN  intsim,log_error;
    BOOLEAN  cir3ptflag ,complex,edmode,squareflag;
    INTEGER ARRAY seg_opcode_table(1:600,1:2);
    INTEGER pict_counter;
    INTEGER prgm_counter;
    INTEGER opcode;
    INTEGER row,segno;
    INTEGER  xlast,ylast;
    INTEGER ARRAY ed_lineopno(1:100,1:2);
    BOOLEAN post_flag ;
```

```
REAL xinc,yinc,lmx,lmy;
REAL sca_d_x,sca_d_y,sca_ch_x,sca_ch_y;
REAL tempsizex,tempsizey;
REAL s1,s2,s3,s4,s5,s6,s7,s8;
REAL shiftx,shifty;
REAL ARRAY pict_table(1:20,1:10);
REAL ARRAY attributes(1:600,1:6);
REAL d_to_r,r_to_d,arc_angle;
REAL chvxl,chvxh,chvyl,chvyt;
REAL ARRAY matrix_trform(1:3,1:3);
REAL ARRAY ed_matform(1:3,1:3);
REAL tx,ty;
REAL wxrc,wxlc,wyrc,wylc;
REAL ARRAY oldptmatrix(1:3);
REAL ARRAY resultmatrix(1:3);
REAL ARRAY temp(1:3,1:3);
REAL ARRAY ed_temp(1:3,1:3);
REF(wi_ndow)window;REF(vi_ewport)viewport;
REF(wv_constant)wvconstant;REF(cl_ipconstant)clipconstant;
REF(ol_ddvst)olddvst;REF(ol_dpolar)oldpolar;
REF(pr_evious)previous;
REAL eddx,eddy,edmx,edmy;
REF(e_d_status)ed_status;
REF(t_tyin)intty;
INTEGER ed_prgmcounter,c_ed_prgmcounter,ed_pictno;
BOOLEAN  edit,re_drawing;
REAL  ARRAY  temp_ed_attr(1:600,1:2),ed_attr(1:600,1:2);
INTEGER ARRAY ed_no_opcode(1:600,1:2),temp_ed_no_opcode
(1:600,1:2);
CHARACTER cha;
TEXT tcr;
PROCEDURE abort(message);
VALUE message;TEXT message;
BEGIN
    IF message =/= NOTEXT THEN
    BEGIN
        Outimage;
        Outtext
        ("EXECUTION TERMINATED DUE TO CALL OF ERROR PROCEDURE");

        Outimage;Outtext("MESSAGE:")
        ;Outtext(message);Outimage;
    END;
END;
PROCEDURE clipstatus(n);INTEGER n;
BEGIN
    opcode:=47;
    IF Open THEN transfer(0,0,0,0,0,0) ELSE
    BEGIN
        IF n=0 THEN clip_flag:=FALSE ELSE
        clip_flag:=TRUE;
```

```
    END**GSIM COMMAND**;
END;
PROCEDURE iden;
BEGIN
    INTEGER i,j;
    FOR i:=1 STEP 1  UNTIL 3 DO
    FOR j:=1 STEP 1 UNTIL 3 DO  matrix_trform(i,j):=0.0;
    matrix_trform(1,1):=matrix_trform(2,2):=
    matrix_trform(3,3):=1.0;
    oldptmatrix(1):=oldptmatrix(2):=0.0;oldptmatrix(3):=1.0;
END;
PROCEDURE   exit;
BEGIN
    opcode:=0;
    IF Open THEN transfer(0,0,0,0,0,0)
    ELSE IF edit THEN  ed_iden ELSE iden;
END**GSIM COMMAND,INITIALISES THE TRANSFORMATION MATRIX**;
PROCEDURE newpage;
BEGIN
    rubout;slow;
END**GSIM COMMAND,CLEARS THE SCREEN**;
PROCEDURE bell;
BEGIN
    opcode:=2;IF Open THEN transfer(0,0,0,0,0,0) ELSE  BEGIN
        Outchar(Char(7));
    END;
END**GSIM COMMAND,RINGS THE BELL IN TTY**;
PROCEDURE  matmul(x,y);REAL ARRAY x;REAL ARRAY y;
BEGIN
    INTEGER i;
    resultmatrix(1):=0.0;resultmatrix(2):=0.0;
    resultmatrix(3):=0.0;

    FOR i:=1 STEP 1 UNTIL 3 DO
    BEGIN
        resultmatrix(1):=x(i)*y(i,1)+resultmatrix(1);
        resultmatrix(2):=x(i)*y(i,2)+resultmatrix(2);
        resultmatrix(3):=x(i)*y(i,3)+resultmatrix(3);
    END;
END;
PROCEDURE trf_mul;
BEGIN
    REAL ARRAY tpy(1:3,1:3);
    INTEGER i,j,k;
    FOR i:=1 STEP 1 UNTIL 3 DO
    FOR k:=1 STEP 1 UNTIL 3 DO
    BEGIN
        tpy(i,k):=0.0;
        FOR j:=1 STEP 1 UNTIL 3 DO
        tpy(i,k):=tpy(i,k)+matrix_trform(i,j)*temp(j,k);
    END;
```

```
    FOR i:=1 STEP 1 UNTIL 3 DO
    FOR j:=1 STEP 1 UNTIL 3 DO
    matrix_trform(i,j):=tpy(i,j);
END;
PROCEDURE translate(ptx,pty);REAL ptx,pty;
BEGIN
    BOOLEAN transflag;
    REAL x,y,tx,ty;
    intty.r1:=tx:=ptx;intty.r2:=ty:=pty;
    IF NOT master THEN BEGIN opcode:=3;ints_status;END;
    transflag:=TRUE;
    temp(1,1):=temp(2,2):=temp(3,3):=1.0;
    WHILE NOT((ed_status.satisfied AND ed_status.count=2)
    OR master)
    OR transflag DO   BEGIN
        IF NOT master THEN       ed_iden;
        transflag:=FALSE;tx:=intty.r1;ty:=intty.r2;
        temp(3,1):=tx; temp(3,2):=ty;
        temp(1,2):=temp(1,3):=temp(2,1):=temp(2,3):=0.0;
        trf_mul;
        IF edit THEN BEGIN
            IF master THEN BEGIN
                ed_temp(1,1):=ed_temp(2,2):=
                ed_temp(3,3):=1.0;

                ed_temp(3,1):=tx;ed_temp(3,2):=ty;
                ed_temp(1,2):=ed_temp(1,3):=
                ed_temp(2,1):=ed_temp(2,3):=0;
                ed_trfmul;
            END ELSE    interact;
        END;
    END;
END**GSIM COMMAND,TRANSLATES THE PICTURE
TO THE SPECIFIED COORDINATE**;
PROCEDURE rotateo(pangle);REAL pangle;
BEGIN
    REAL y;
    REAL angle;BOOLEAN rotateoflag;
    IF NOT master THEN BEGIN ints_status;opcode:=4;
    END;rotateoflag:=TRUE;
    intty.r1:=angle:=pangle;
    WHILE NOT((ed_status.satisfied AND ed_status.count=2)
    OR master)                OR rotateoflag DO BEGIN

        IF NOT master THEN                  ed_iden;
        rotateoflag:=FALSE;angle:=intty.r1;
        temp(1,1):=temp(2,2):=Cos(angle*d_to_r);
        temp(2,1):=Sin(angle*d_to_r);
        temp(1,2):=-Sin(angle*d_to_r);
        temp(1,3):=temp(2,3):=temp(3,1):=temp(3,2):=0.0;
        temp(3,3):=1.0;
```

```
        trf_mul;
        IF edit THEN  BEGIN
            IF master THEN BEGIN
                ed_temp(1,3):=ed_temp(2,3):=
                ed_temp(3,1):=ed_temp(3,2):=0;
                ed_temp(1,1):=ed_temp(2,2):=
                Cos(angle*d_to_r);
                ed_temp(2,1):=Sin(angle*d_to_r);
                ed_temp(1,2):=-Sin(angle*d_to_r)
                ;ed_temp(3,3):=1.0;
                ed_trfmul;
            END ELSE interact;
        END;
    END;
END**ROTATES THE  PICTURE WITH RESPECT
TO THE ORIGIN BY THE GIVEN ANGLE**;
PROCEDURE rotatept(pq,px,py);REAL pq,px,py;
BEGIN
    BOOLEAN rotateptflag;
    REAL q,x,y;
    master:=TRUE;opcode:=5;intty.r1:=q:=pq;
    ints_status;
    intty.r2:=x:=px;intty.r3:=y:=py;rotateptflag:=TRUE;
    WHILE NOT(ed_status.satisfied AND ed_status.count=2)
    OR rotateptflag DO BEGIN
        ed_iden;
        rotateptflag:=FALSE;
        q:=intty.r1;x:=intty.r2;y:=intty.r3;
        translate(-x,-y);
        rotateo(q);
        translate(x,y);
        IF edit THEN             interact;
    END;
    master:=FALSE;
END** ROTATES THE PICTURE WITH RESPECT
TO THE POINT PX,PY BY AN ANGLE PQ**;
PROCEDURE fill_table;
BEGIN
    pict_table(pict_counter,1):=segno;
    pict_table(pict_counter,2):=matrix_trform(1,1);
    pict_table(pict_counter,3):=matrix_trform(1,2);
    pict_table(pict_counter,4):=matrix_trform(1,3);
    pict_table(pict_counter,5):=matrix_trform(2,1);
    pict_table(pict_counter,6):=matrix_trform(2,2);
    pict_table(pict_counter,7):=matrix_trform(2,3);
    pict_table(pict_counter,8):=matrix_trform(3,1);
    pict_table(pict_counter,9):=matrix_trform(3,2);
    pict_table(pict_counter,10):=matrix_trform(3,3);
    pict_counter:=pict_counter+1;
END;
PROCEDURE open_segment(x);INTEGER x;
```

```
BEGIN
    IF Open THEN BEGIN Outimage;
        Outtext("GSIM ERROR # 1;");
        Outimage;
        GO TO done;
        GOTO done;
    END
    ELSE  IF edit THEN BEGIN
        Outimage;
    Outtext(" GSIM ERROR #3 ");GO TO done;Outimage END ELSE
    BEGIN
        opcode:=6;
        Open:=TRUE;
        segno:=x;
    END;
END;

PROCEDURE close_segment(x);INTEGER  x;
BEGIN
    IF x \= segno THEN BEGIN
        Outimage;
        Outtext("GSIM ERROR # 4;;");
        Outimage;
        GO TO done;
    END ELSE
    IF NOT Open THEN BEGIN
        Outimage;
        Outtext("GSIM  ERROR # 4;;");
        Outimage;
        GO TO done;
    END
    ELSE
    BEGIN
        opcode:=7;
        Open:=FALSE;
        fill_table;
    END
END;
PROCEDURE init_buff(x);INTEGER x;
BEGIN
    INTEGER i,j,k;
    iden;IF NOT(Open AND edit) THEN BEGIN
        IF x=0 THEN  k:=100 ELSE
        IF x=1 THEN k:=200 ELSE
        k:=600;
        FOR j:=1 STEP 1 UNTIL 6 DO
        FOR i:=1 STEP 1 UNTIL k DO
        attributes(i,j):=0.0;
    END;
END;
PROCEDURE transfer(t1,t2,t3,t4,t5,t6);REAL t1,t2,t3,t4,t5,t6;
```

```
BEGIN
    seg_opcode_table(prgm_counter,1):=segno;
    seg_opcode_table(prgm_counter,2):=opcode;
    attributes(prgm_counter,1):=t1;
    attributes(prgm_counter,2):=t2;
    attributes(prgm_counter,3):=t3;
    attributes(prgm_counter,4):=t4;
    attributes(prgm_counter,5):=t5;
    attributes(prgm_counter,6):=t6;
    prgm_counter:=prgm_counter+1;
END;
PROCEDURE moveto(px,py);REAL px,py;
BEGIN
    REAL ARRAY st(1:3);REAL  p,q,x,y;BOOLEAN movetoflag;
    BOOLEAN counter;
    IF  NOT intsim THEN BEGIN
        Outtext("GSIM  ERROR  #6;;");
        Outimage;GO TO done;
    END
    ELSE  BEGIN
        movetoflag:=TRUE;
        IF Open THEN  opcode:=9 ELSE
        IF edit THEN BEGIN
            IF (ed_status.satisfied AND ed_status.count=1)
            THEN  BEGIN opcode:=9 ;counter:=TRUE;
            END ELSE
            IF NOT complex THEN  BEGIN
                ints_status;opcode:=9;
                counter:=FALSE;
            END ELSE counter:=TRUE
        END ELSE BEGIN
            IF NOT complex THEN BEGIN
                ints_status;opcode:=9;counter:=
                FALSE;
        END ELSE counter:=TRUE;END;
        intty.r1:=x:=px;intty.r2:=y:=py;
        IF Open THEN transfer(x,y,0.0,0.0,0.0,0.0) ELSE BEGIN
            WHILE NOT
            ((ed_status.satisfied AND ed_status.count=2)  OR
            counter)
            OR                  movetoflag DO  BEGIN
                movetoflag:=FALSE;x:=intty.r1;
                y:=intty.r2;
                IF post_flag THEN
                BEGIN x:=x+shiftx;
                y:=y+shifty;END;
                st(1):=x;st(2):=y;st(3):=1;
                matmul(st,matrix_trform);
                p:=resultmatrix(1);
                q:=resultmatrix(2);
                IF ed_status.count=1 THEN BEGIN
```

```
                    editedxy_save(x,y);
                    previous.tx:=olddvst.cx
                    :=wvconstant.wvxm*p+wvconstant.wvxa;
                    previous.ty:=olddvst.cy
                    :=wvconstant.wvym*q+wvconstant.wvya;
                END
                ELSE
                BEGIN
                    previous.tx:=olddvst.cx
                    :=wvconstant.wvxm*p+wvconstant.wvxa;
                    previous.ty:=olddvst.cy
                    :=wvconstant.wvym*q+wvconstant.wvya;
                    IF (edit AND NOT complex
                    ) THEN BEGIN
                        ed_showline(olddvst.cx,olddvst.cy);
                        IF NOT re_drawing THEN
                    Inimage;END;
                END;
                IF ((NOT complex)  AND edit )
            THEN    interact;END;
        END;
    END;
END;
PROCEDURE set_w_v_constants;
BEGIN
    wvconstant.wvxm:=(viewport.vxr-viewport.vxl)/
    (window.wxr-window.wxl);
    wvconstant.wvxa:=(viewport.vxl-window.wxl*
    wvconstant.wvxm);
    wvconstant.wvym:=(viewport.vyt-viewport.vyb)/
    (window.wyt-window.wyb);
    wvconstant.wvya:=(viewport.vyb-window.wyb*
    wvconstant.wvym);
END;
PROCEDURE setwindow(wx1,wx2,wy1,wy2);
REAL wx1,wx2,wy1,wy2;
BEGIN
    REAL x1,x2,y1,y2;BOOLEAN setwflag;
    opcode:=10;setwflag:=TRUE;
    intty.r1:=x1:=wx1;intty.r2:=x2:=wx2;intty.r3:=y1:=wy1;
    intty.r4:=y2:=wy2;log_error:=TRUE;
    WHILE log_error DO
    IF ( intty.r1 >= intty.r2) OR (intty.r3 >= intty.r4)
    THEN BEGIN
        Outimage;


        IF intty.r1>=intty.r2 THEN Outtext
        (" XLOW IS >= XHIGH");
        Outimage;IF intty.r3>=intty.r4 THEN
        Outtext("YLOW IS >= YHIGH");
```

```
    interact; END ELSE log_error:=FALSE;
    ints_status;
    IF Open THEN transfer(x1,x2,y1,y2,0.0,0.0,0.0) ELSE BEGIN
        WHILE NOT(ed_status.satisfied  AND
        ed_status.count=2) OR setwflag
        DO BEGIN
            setwflag:=FALSE;
            x1:=intty.r1;x2:=intty.r2;y1:=intty.r3
            ;y2:=intty.r4;
            window.wx1:=x1;window.wxr:=x2;
            window.wyb:=y1;window.wyt:=y2;
            wxrc:=x2;wx1c:=x1;wyrc:=y2;wy1c:=y1;
            set_w_v_constants;
            increment;
            IF edit THEN              interact;
        END;
    END;
END;
PROCEDURE ed_showline(p,q);REAL p,q;
BEGIN
    Outchar(Char(29));
    transmit_coordinates(p,q);
    Outchar(Char(31));
END;
PROCEDURE setviewport(vx1,vx2,vy1,vy2);
REAL vx1,vx2,vy1,vy2;
BEGIN
    REAL x1,x2,y1,y2;BOOLEAN setvflag;
    opcode:=11;
    setvflag:=TRUE;intty.r1:=x1:=vx1;intty.r2:=x2:=vx2;
    intty.r3:=y1:=vy1;intty.r4:=y2:=vy2;
    log_error:=TRUE;
    WHILE log_error DO
    IF (intty.r1 >= intty.r2) OR
    (intty.r3 >=intty.r4) THEN
    BEGIN
        Outimage;IF intty.r1>=intty.r2 THEN
        Outtext("XLOW IS >= XHIGH");
        Outimage;IF intty.r3>=intty.r4 THEN Outtext
        ("YLOW >= YHIGH");interact;
    END
    ELSE log_error:=FALSE;
    ints_status;
    IF Open THEN transfer(x1,x2,y1,y2,0.0,0.0,0.0) ELSE BEGIN
        WHILE NOT(ed_status.satisfied AND
        ed_status.count=2) OR setvflag
        DO BEGIN
            setvflag:=FALSE;chvx1:=x1:=
            intty.r1;chvxh:=x2:=intty.r2;
            chvy1:= y1:=intty.r3;chvyt:=y2:=
            intty.r4;
```

```
                clipconstant.clipxl:=viewport.vxl
                :=x1+240;
                clipconstant.clipxr:=viewport.vxr
                :=x2+240;
                clipconstant.clipyb:=viewport.vyb:=y1;
                clipconstant.clipyt:=viewport.vyt:=y2;
                set_w_v_constants;
                increment;
                IF edit THEN            interact;
            END;
        END;
    END;
    PROCEDURE slow;
    BEGIN
        INTEGER i,j;
        j:=1;FOR i:=1 STEP 1 UNTIL 50000 DO
        j:=j*1;
    END;
    PROCEDURE transmit_coordinates(x,y);
    REAL x,y;
    BEGIN
        INTEGER xi,yi;
        xi:=x//1;yi:=y//1;
        Outchar(Char(yi//32+32));
        Outchar(Char(Mod(yi,32)+96));
        Outchar(Char(xi//32+32));
        Outchar(Char(Mod(xi,32)+64));
        Breakoutimage;
    END;
    PROCEDURE show_line(x1,y1,x2,y2);
    REAL x1,y1,x2,y2;
    BEGIN
        Outchar(Char(29));
        transmit_coordinates(x1,y1);
        transmit_coordinates(x2,y2);
        Outchar(Char(31));
    END;
    PROCEDURE clip(x1,y1,x2,y2);REAL x1,y1,x2,y2;
    BEGIN
        CLASS position;
        BEGIN
            BOOLEAN left,right,bottom,top;
        END;
        REAL x,y;
        REF(position)c,c1,c2;
        PROCEDURE code(x,y);
        REAL x,y;
        BEGIN
            INSPECT c DO BEGIN left:=right:=
            top:=bottom:=FALSE;END;
            IF x < clipconstant.clipxl  THEN
```

```
    c.left:=TRUE   ELSE
    IF x > clipconstant.clipxr   THEN c.right:=TRUE;
    IF y < clipconstant.clipyb  THEN c.bottom
    :=TRUE  ELSE
    IF y >clipconstant.clipyt   THEN c.top:=TRUE;
END;
IF clip_flag THEN
BEGIN
    c:-NEW position;code(x1,y1);    c1:-c;
    c:-NEW position;code(x2,y2);    c2:-c;
    WHILE  (c1.left OR c1.right OR c1.bottom
    OR c1.top)  OR
    (c2.left OR c2.right OR c2.bottom OR c2.top)  DO
    BEGIN
        IF  ((c1.left AND c2.left) OR (c1.right
        AND c2.right)
        OR  (c1.bottom AND c2.bottom) OR
        (c1.top AND c2.top))  THEN
        GO TO return;
        c:-c1;
        IF NOT (c.left OR c.right OR c.bottom
        OR c.top)  THEN  c:-c2;
        IF c.left  THEN
        BEGIN
            y:=y1+(y2-y1)*(clipconstant.
            clipx1-x1)/(x2-x1);
            x:=clipconstant.clipx1;
        END ELSE
        IF c.right THEN
        BEGIN
            y:=y1+(y2-y1)*(clipconstant.
            clipxr-x1)/(x2-x1);
            x:=clipconstant.clipxr;
        END ELSE
        IF c.bottom THEN
        BEGIN
            x:=x1+(x2-x1)*(clipconstant.
            clipyb-y1)/(y2-y1);
            y:=clipconstant.clipyb;
        END ELSE
        IF c.top THEN BEGIN
            x:=x1+(x2-x1)*(clipconstant.
            clipyt-y1)/(y2-y1);
            y:=clipconstant.clipyt;
        END;
        IF c==c1 THEN    BEGIN
            x1:=x;
            y1:=y;
            code(x,y);
        END  ELSE
        BEGIN
```

```
                    x2:=x;y2:=y;code(x,y);
                    c2:=c;
                END;
            END;
        END;
        show_line(x1,y1,x2,y2);
return:END;
PROCEDURE drawto(px,py);
REAL  px,py;
BEGIN
        REAL nx,ny,x,y;BOOLEAN drawtoflag,counter;
        REAL ARRAY st(1:3);REAL p,q;
        IF NOT intsim  THEN BEGIN
            Outimage;
            Outtext(" GSIM ERROR # 6;;");
            Outimage;GO TO done;
        END
        ELSE BEGIN
            IF Open THEN opcode:=12  ELSE
            IF edit THEN BEGIN
                IF (ed_status.satisfied AND ed_status
                .count=1) THEN
                BEGIN
                opcode:=12;  counter:=TRUE;  END ELSE
                IF NOT complex THEN BEGIN
                ints_status;opcode:=12;END ELSE
                counter:=TRUE
            END ELSE BEGIN
                IF NOT complex THEN BEGIN
                    ints_status;opcode:=12;
                    counter:=FALSE;
                END
                ELSE counter:=TRUE;
            END;
            intty.r1:=x:=px;intty.r2:=y:=py;drawtoflag:=TRUE;
            IF Open THEN transfer(x,y,0.0,0.0,0.0,0.0)
            ELSE   BEGIN
                WHILE NOT((ed_status.satisfied AND
                ed_status.count=2) OR
                counter ) OR drawtoflag DO BEGIN
                    x:=intty.r1;y
                    :=intty.r2;drawtoflag:=FALSE;
                    IF post_flag THEN
                    BEGIN
                        x:=x+shiftx;y:=y+shifty;
                    END;
                    st(1):=x;st(2):=y;st(3):=1.0;
                    matmul(st,matrix_trform);
                    p:=resultmatrix(1);
                    q:=resultmatrix(2);
                    nx:=wvconstant.wvxm*
```

```
                    p+wvconstant.wvxa;
                    ny:=wvconstant.wvym*q
                    +wvconstant.wvya;
                    IF ed_status.count=1 THEN
                    editedxy_save(x,y) ELSE
                    BEGIN
                        IF ( (NOT edit) OR
                        ed_status.count\= 1 ) THEN
                        clip(olddvst.cx,olddvst.
                        cy,nx,ny);
                        BEGIN
                            olddvst.cx:=
                            nx;olddvst.cy:=ny;
                            previous.
                            tx:=nx;previous.ty:=ny;
                        END;
                    END;
                    IF NOT complex THEN
                    interact;
                END;
            END;
        END;
    END;
PROCEDURE init_gsim;
BEGIN
    Open:=FALSE;edit:=FALSE;
    IF NOT(Open AND edit) THEN BEGIN
        ed_status:=NEW e_d_status;
        window:=NEW  wi_ndow;
        viewport:=NEW    vi_ewport;
        wvconstant:=NEW  wv_constant;
        clipconstant:=NEW  cl_ipconstant;
        intty:=NEW t_tyin;
        clipstatus(1);
        olddvst:=NEW  ol_ddvst;
        oldpolar:=NEW ol_dpolar;
        previous:=NEW pr_evious;
        setwindow(0.0,780.0,0.0,780.0);
        setviewport(0.0,780.0,0.0,780.0);
        Open:=FALSE;ed_iden;ints_status;
        clip_flag:=TRUE;
        post_flag:=FALSE;
        prgm_counter:=1;
        pict_counter:=1;
        edit:=FALSE;re_drawing:=FALSE;
        ed_prgmcounter:=c_ed_prgmcounter:=0;
        iden;
        slow;
        intsim:=TRUE;
    END;
END;
```

```
PROCEDURE circle(px,py,pr);
REAL px,py,pr;
BEGIN
    REAL xr,yr,d,angle,x,y,r;
    INTEGER i;BOOLEAN cirflag;
    intty.r1:=x:=px;intty.r2:=y:=py;intty.r3:=r:=pr;
    cirflag:=TRUE;complex:=TRUE;
    IF NOT  master THEN  BEGIN opcode:=14;
        log_error:=TRUE;
    END;

    WHILE log_error  AND NOT master DO
    BEGIN
        IF intty.r3 <= 0 THEN
        BEGIN
            Outimage;
            Outtext(" RADIUS IS -VE OR 0");
        Outimage;interact; END ELSE
        log_error:=FALSE;
    END;
    ints_status;
    WHILE NOT((ed_status.satisfied AND ed_status.count
    =2) OR cir3ptflag ) OR
    cirflag  DO
    BEGIN
        cirflag:=FALSE;
        IF NOT  master   THEN                  complex:=TRUE;
        d:=d_to_r;
        x:=intty.r1;y:=intty.r2;r:=intty.r3;xr:=r;yr:=0;
        moveto(x+xr,y+yr);
        FOR i:=1 STEP 1 UNTIL 360 DO
        BEGIN
            angle:=i*d;
            xr:=r*Cos(angle);
            yr:=r*Sin(angle);
            drawto(x+xr,y+yr);
        END;
        intty.r1:=x;intty.r2:=y;intty.r3:=r;
        IF NOT  master  THEN BEGIN IF edit
        THEN interact;END;
    END;
    IF NOT  master  THEN complex:=FALSE;

END;
PROCEDURE rectangle(px,py,pp,pq);
REAL px,py,pp,pq;
BEGIN
    REAL x,y,p,q;BOOLEAN rectflag;
    IF NOT master THEN BEGIN   opcode:=15;ints_status;
    END;intty.r1:=x:=px;
```

```
    intty.r2:=y;*py;intty.r3:=p;*pp;intty.r4:=q;*pq;
    rectflag:=TRUE;
    IF NOT master THEN
    BEGIN
        log_error:=TRUE;
        WHILE log_error  AND NOT  master  DO
        IF (( intty.r3 <= 0)  OR (
        intty.r4
        <= 0))
        THEN   BEGIN
            Outimage;
            Outtext("SIDE/S OF RECTANGLE IS 0/-VE!");
        Outimage;interact;END ELSE log_error:=FALSE;
        ints_status;
    END;
    WHILE NOT ((ed_status.satisfied  AND  ed_status.count=2)
    OR squareflag)
    OR rectflag DO BEGIN
        complex:=TRUE;rectflag:=FALSE;
        x:=intty.r1;y:=intty.r2;p:=intty.r3;q:=intty.r4;
        moveto(x,y);x:=x+p;
        drawto(x,y);y:=y+q;
        drawto(x,y);x:=x-p;
        drawto(x,y);y:=y-q;
        drawto(x,y);
        intty.r1:=x;intty.r2:=y;intty.r3:=p;intty.r4:=q;
        IF NOT master  THEN BEGIN IF edit
        THEN                interact;END;
    END;
    IF NOT  master  THEN    complex:=FALSE;
END;
PROCEDURE square(px,py,pl);REAL px,py,pl;
BEGIN
    REAL x,y,p;
    x:=px;y:=py;p:=pl;ints_status;
    squareflag:=TRUE;opcode:=16;
    log_error:=TRUE;complex:=TRUE;master:=TRUE;
    WHILE log_error DO BEGIN
        IF p <= 0 THEN BEGIN
            Outtext("ERROR IN DATA  SQUARE")
            ;Outimage;
            Outfix(x,2,5);Outfix(y,2,5);
            Outfix(p,2,5);Outimage;
            Outtext("GIVE CORRECT VALUES");
            x:=Inreal;y:=Inreal;p:=Inreal;
            Inimage;log_error:=TRUE;
        END
        ELSE log_error:=FALSE;
    END;
    repeat1:
    complex := TRUE;rectangle(x,y,p,p);
```

```
    IF ed_status.count \= 1 THEN BEGIN
        IF edit THEN BEGIN
            Outtext("Y TO MODIFY ELSE N");Outimage;
            Inimage;ed_status.ch:=Inchar;
            IF ed_status.ch ='Y' THEN  BEGIN
                Outtext("  SQUARE   ");Outimage;
                Outimage;
                Outtext("  GIVE NEW VALUES  ");
                Outimage;x:=Inreal;y:=Inreal;
                p:=Inreal;
                re_draw_edited;
            END ELSE
            BEGIN
                ed_status.satisfied:=TRUE;
                ed_status.count:=0;
            END;
            ed_status.count:=ed_status.count+1;
        END
        ELSE  ed_status.count:=2;
    END ELSE ed_status.count:=2;
    IF  NOT (ed_status.satisfied AND ed_status.count=2)
    THEN  GOTO repeat1;
    squareflag:=FALSE;complex:=FALSE;master:=FALSE;
END;
PROCEDURE ints_status;
BEGIN
    ed_status.ch:='Z';ed_status.count:=2;
    IF edit THEN ed_status.satisfied:=FALSE  ELSE
    ed_status.satisfied:=TRUE;
END;
PROCEDURE showchar(c,x,y);
CHARACTER  c;REAL x,y;
BEGIN
    IF NOT intsim THEN BEGIN
        Outimage;
        Outtext("GSIM ERROR # 6;;");
        Outimage;GO TO done;
    END ELSE BEGIN
        Outchar(Char(29));
        transmit_coordinates(x,y);
        Outchar(Char(31));
        Outchar(Char(Rank(c)));
    END;
END;
PROCEDURE drawchar(c);
CHARACTER  c;
BEGIN
    INTEGER textwidth,textheight;
    textwidth:=20;textheight:=20;
    opcode:=17;
    IF Open THEN      BEGIN       END ELSE BEGIN
```

```
        IF (olddvst.cx>=clipconstant.clipx1)  AND
        (olddvst.cx+textwidth-1 <=
        clipconstant.clipxr) AND
        (olddvst.cy>= clipconstant.clipyb) AND
        (olddvst.cy+textwidth-1<=clipconstant.clipyt)
        THEN  BEGIN
            showchar(c,olddvst.cx,olddvst.cy);
            olddvst.cx:=olddvst.cx+textwidth;
        END;
    END;
END;
PROCEDURE drawtext(s);
TEXT s;
BEGIN
    INTEGER j,  i;CHARACTER  c;
    TEXT t;
    opcode:=18;
    IF Open THEN BEGIN     END  ELSE BEGIN

        t:=s;
        j:=t.Length;
        t.Setpos(1);
        FOR i:=1 STEP 1 UNTIL j  DO
        BEGIN
            c:=t.Getchar;drawchar(c);
        END;
    END;
END;
PROCEDURE graphplot(n,vx,vy,sx,sy);
INTEGER n;INTEGER ARRAY vx,vy;TEXT sx,sy;
BEGIN
    INTEGER w,h;
    INTEGER  xmax,ymax,xinterval,yinterval,i;
    TEXT s;
    w:=15;h:=12;
    BEGIN
        opcode:=19;complex:=TRUE;
        xmax:=vx(1);ymax:=vy(1);
        FOR i:=2 STEP 1 UNTIL n DO
        BEGIN
            IF vx(i)>xmax THEN xmax:=vx(i);
            IF vy(i)>ymax THEN ymax:=vy(i);
        END;
        xinterval:=xmax//10+1;
        xmax:=xinterval*10;
        yinterval:=ymax//10+1;
        ymax:=yinterval*10;
        setwindow(0,xmax,0,ymax);
        setviewport(6*w,780,4*h,780);
        moveto(0,ymax);drawto(0,0);drawto(xmax,0);
        moveto(vx(1),vy(1));
```

```
        FOR i:=2 STEP 1 UNTIL n DO drawto(vx(1),vy(1));
        setwindow(0,xmax,0,4*h);
        setviewport(6*w,780,0,4*h);
        moveto(xinterval,0);
        drawtext(sx);
        FOR i:=1 STEP 1 UNTIL 9 DO
        BEGIN
            moveto(xinterval*i,4*h);
            drawto(xinterval*i,3*h);
            moveto(xinterval*i-2*w,.1*h);
            Outint(i,2);
        END;
        setwindow(0,6*w,0,ymax);setviewport(0,6*w,
        4*h,780);
        moveto(0,9.5*yinterval);drawtext(sy);
        FOR i:=1 STEP 1 UNTIL 9 DO
        BEGIN
            moveto(6*w,yinterval*i);
            drawto(5*w,yinterval*i);
            moveto(0,yinterval*i);
            Outint(i,2);
        END;
    END;
    complex:=FALSE;
END;
PROCEDURE poly(n,vx,vy);
INTEGER n;
REAL ARRAY vx,vy;
BEGIN
    INTEGER i,j;
    log_error:=TRUE;
    WHILE log_error DO
    BEGIN
        IF n<=0 THEN BEGIN
            Outimage;Outtext("ERROR IN DATA");
            Outimage;Outtext("FOR POLY");Outimage;
            Outtext("N IS -VE OR 0");
            Outimage;Outtext("GIVE CORRECT VALUE");
        Outimage;n:=Inint;END
        ELSE log_error:=FALSE;
    END;
    ints_status;opcode:=20;
    complex:=TRUE;
    moveto(vx(1),vy(1));
    i:=1;
    WHILE i<n DO BEGIN
        drawto(vx(i+1),vy(i+1));
        i:=i+1;
    END;
    drawto(vx(1),vy(1));
    complex:=FALSE;
```

```
END;
PROCEDURE rmoveto(px,py);
REAL  px,py;
BEGIN
     REAL x1,y1,x,y;BOOLEAN rmovetoflag;
     IF NOT intsim THEN BEGIN
          Outimage;
          Outtext("GSIM ERROR # 6;;");
          Outimage;GO TO done;
     END ELSE BEGIN
          opcode:=21;
          intty.r1:=x:=px;intty.r2:=y:=py;
          rmovetoflag:=TRUE;
          ints_status;complex:=TRUE;
          WHILE NOT(ed_status.satisfied AND
          ed_status.count=2) OR
          rmovetoflag DO BEGIN
               rmovetoflag:=FALSE;x:=intty.r1;y:=
               intty.r2;
               x1:=(olddvst.cx-wvconstant.wvxa)/
               wvconstant.wvxm+x;
               y1:=(olddvst.cy-wvconstant.wvya)/
               wvconstant.wvym+y;
               moveto(x1,y1);
               intty.r1:=x;intty.r2:=y;
               IF edit THEN interact;
          END;
     END;
     IF NOT master    THEN complex:=FALSE;
END;
PROCEDURE rdrawto(px,py);
REAL px,py;
BEGIN
     REAL x1,y1,x,y;BOOLEAN rdrawtoflag;
     IF NOT intsim THEN BEGIN
          Outimage;
          Outtext("GSIM ERROR # 6;;");
          Outimage;GO TO done;
     END ELSE BEGIN
          opcode:=22;
          intty.r1:=x:=px;intty.r2:=y:=py;
          rdrawtoflag:=TRUE;ints_status;
          complex:=TRUE;
          WHILE NOT(ed_status.satisfied AND
          ed_status.count=2)
          OR rdrawtoflag

          DO BEGIN
               rdrawtoflag:=FALSE;
               x1:=(olddvst.cx-wvconstant.wvxa)/
               wvconstant.wvxm+x;
```

```
            y1:=(olddvst.cy-wvconstant.wvya)/
            wvconstant.wvym+y;
            drawto(x1,y1);intty.r1:=x;
            intty.r2:=y;IF edit THEN interact;
        END;
    END;
    IF NOT master    THEN complex:=FALSE;
END;

PROCEDURE mpolar(pr,pangle);
REAL pr,pangle;
BEGIN
    REAL x,y,theta,r,angle;BOOLEAN mpolflag;
    intty.r1:=r:=pr;intty.r2:=angle:=pangle;
    opcode:=23;mpolflag:=TRUE;complex:=TRUE;
    log_error:=TRUE;
    WHILE log_error DO
    IF  intty.r1 <=0 THEN  interact ELSE log_error:=FALSE;
    ints_status;
    WHILE NOT (ed_status.satisfied AND ed_status.count=2)
    OR mpolflag DO BEGIN

        mpolflag:=FALSE;r:=intty.r1;angle:=intty.r2;
        theta:=angle*(d_to_r);
        x:=r*Cos(theta);
        y:=r*Sin(theta);
        moveto(x,y);
        oldpolar.rlast:=r;oldpolar.alast:=angle;
        intty.r1:=r;intty.r2:=angle;
        IF edit THEN              interact;
    END;
    complex:=FALSE;
END;
PROCEDURE dpolar(pr,pangle);
REAL pr,pangle;
BEGIN
    REAL r,angle;
    BOOLEAN  dpolflag;
    INTEGER i,j;
    REAL a1, delr,r1,theta,dela;
    REAL x,y;
    complex:=TRUE;
    intty.r1:=r:=pr;intty.r2:=angle:=pangle;
    dpolflag:=TRUE;opcode:=24;
    log_error:=TRUE;
    WHILE log_error DO IF intty.r1 <=0 THEN
    interact ELSE log_error:=FALSE;
    ints_status;
    WHILE NOT (ed_status.satisfied AND ed_status.count=2)
    OR dpolflag DO BEGIN
        dpolflag:=FALSE;
```

```
        r:=intty.r1;angle:=intty.r2;
        delr:=(r-oldpolar.rlast)/360;
        dela:=(angle-oldpolar.alast)/360;
        r1:=oldpolar.rlast;a1:=oldpolar.alast;
        FOR j:=1 STEP 1 UNTIL 360 DO
        BEGIN
            r1:=r1+delr;a1:=a1+dela;
            theta:=a1*d_to_r;
            x:=r1*Cos(theta);
            y:=r1*Sin(theta);
            drawto(x,y);
            oldpolar.rlast:=r;
            oldpolar.alast:=angle;
        END;
        intty.r1:=r;intty.r2:=angle;
        IF edit THEN                interact;
    END;

    complex:=FALSE;
END;
PROCEDURE arc(pp,pstangle,pendangle);
REAL pstangle,pendangle;
REAL pp;
BEGIN
    INTEGER i,n;BOOLEAN arcflag;
    REAL x1,y1,xr,yr,angle,m,p,stangle,endangle;
    complex:=TRUE;arcflag:=TRUE;
    intty.r1:=p:=pp;intty.r2:=stangle:=pstangle;
    intty.r3:=endangle:=pendangle;
    IF NOT master THEN
    BEGIN
        ints_status;          log_error:=TRUE;opcode:=25;
    END;
    WHILE log_error   AND NOT master DO
    IF intty.r1 <= 0 THEN
    interact ELSE log_error:=FALSE;
    WHILE NOT((ed_status.satisfied AND
    ed_status.count=2)  OR master )
    OR arcflag DO BEGIN
        arcflag:=FALSE;p:=intty.r1;stangle:=intty.r2;
        endangle:=intty.r3;


        m:=11/630;
        x1:=(olddvst.cx-wvconstant.wvxa)/wvconstant.wvxm
        -p*Cos(stangle*m);

        y1:=(olddvst.cy-wvconstant.wvya)/wvconstant.wvym
        -p*Sin(stangle*m);
        IF stangle > endangle THEN
        BEGIN
```

```
            n:=360-stangle;
            FOR i:=1 STEP 1 UNTIL n DO
            BEGIN
                angle:=i*d_to_r;
                xr:=p*Cos(angle+stangle*d_to_r);
                yr:=p*Sin(angle+stangle*d_to_r);
                drawto(xr+x1,yr+y1);
            END;
            n:=endangle;
            FOR i:= 1 STEP 1 UNTIL n DO
            BEGIN
                angle:=i*d_to_r;
                xr:=p*Cos(angle);
                yr:=p*Sin(angle);
                drawto(xr+x1,yr+y1);
            END;
        END  ELSE
        BEGIN
            n:=endangle-stangle;
            FOR i:=1 STEP 1 UNTIL n DO
            BEGIN
                angle:=i*m;
                xr:=p*Cos(angle+stangle*m);
                yr:=p*Sin(angle+stangle*m);
                drawto(xr+x1,yr+y1);
            END;
        END;
        intty.r1:=p;intty.r2:=stangle;intty.r3:=endangle;
        IF NOT master THEN
        BEGIN
            IF edit THEN interact
        END;
    END;
    IF NOT master THEN      complex:=FALSE;


END;
PROCEDURE int_temp;
BEGIN
    INTEGER i,j;
    FOR i:=1 STEP 1 UNTIL 3 DO
    FOR j:=1 STEP 1 UNTIL 3 DO
    temp(i,j):=0.0;
    FOR i:=1 STEP 1 UNTIL 3 DO temp(i,i):=1.0;
END;

PROCEDURE shear_x(px);REAL px;
BEGIN
    BOOLEAN shearxflag;REAL x;
    intty.r1:=x:=px;ints_status;shearxflag:=TRUE;
    int_temp;temp(2,1):=x;opcode:=26;
```

```
    WHILE NOT (ed_status.satisfied AND ed_status.count=2) OR
    shearxflag DO
    BEGIN
        ed_iden;shearxflag:=FALSE;x:=intty.r1;int_temp;
        temp(2,1):=x;trf_mul;
        IF edit THEN                interact;
    END;
END;

PROCEDURE shear_y(px);REAL px;
BEGIN
    BOOLEAN shearyflag;REAL x;
    intty.r1:=x:=px;ints_status;shearyflag:=TRUE;
    int_temp;temp(1,2):=x;opcode:=27;
    WHILE NOT (ed_status.satisfied AND ed_status.count=2)
    OR shearyflag DO
    BEGIN
        ed_iden;shearyflag:=FALSE;x:=intty.r1;
        int_temp;temp(1,2):=x;trf_mul;
        IF edit THEN                interact;
    END;
END;
PROCEDURE reflect_xeqy;
BEGIN
    opcode:=110;IF Open THEN transfer(0,0,0,0,0,0) ELSE
    IF edit THEN
    BEGIN
        ed_temp(1,2):=ed_temp(2,1):=1.0;
        ed_temp(1,1):=ed_temp(2,2):=0;
        ed_temp(1,3):=ed_temp(2,3):=ed_temp(3,1):=
        ed_temp(3,2):=0;
        ed_temp(3,3):=1;
        ed_trfmul;
    END
    ELSE
    BEGIN
        int_temp;temp(1,2):=temp(2,1):=1;
        temp(1,1):=temp(2,2):=0;trf_mul;
    END;
END;
PROCEDURE scale_uni(px);REAL px;
BEGIN
    REAL x;BOOLEAN scaleuniflag;
    ints_status;scaleuniflag:=TRUE;intty.r1:=x:=px;
    opcode:=29;
    WHILE NOT(ed_status.satisfied AND ed_status.count=2) OR
    scaleuniflag DO
    BEGIN
        ed_iden;
        scaleuniflag:=FALSE;x:=intty.r1;
        int_temp;
```

```
        temp(1,1):=temp(2,2):=x;
        trf_mul;
        IF edit THEN              interact;
    END;
END;
PROCEDURE scale(ppx,ppy);
REAL ppx,ppy;
BEGIN
    REAL px,py;BOOLEAN scaleflag;
    ints_status;scaleflag:=TRUE;intty.r1:=px:=ppx;
    intty.r2:=py:=ppy;opcode:=29;
    WHILE NOT(ed_status.satisfied AND ed_status.count=2) OR
    scaleflag DO
    BEGIN
        IF NOT master THEN ed_iden;
        scaleflag:=FALSE;px:=intty.r1;py:=intty.r2;
        int_temp;
        temp(1,1):=px;temp(2,2):=py;
        trf_mul;
        IF edit THEN
        BEGIN
            IF master THEN
            BEGIN
                ed_temp(1,1):=px;ed_temp(2,2):=
                py;ed_temp(3,3):=1.0;
                ed_temp(1,2):=
                ed_temp(1,3):=ed_temp(2,1):=
                ed_temp(2,3):=ed_temp(3,1):=0;
                ed_temp(3,2):=0;ed_trfmul;
            END
            ELSE
            interact;
        END;
    END;
END;
PROCEDURE scalept(sx,sy,x,y);REAL sx,sy,x,y;
BEGIN
    REAL psx,psy,px,py;BOOLEAN scaleptflag;
    master:=TRUE;opcode:=30;
    intty.r1:=psx:=sx;intty.r2:=psy:=sy;
    intty.r3:=px:=x;intty.r4:=py:=y;
    scaleptflag:=TRUE;
    WHILE NOT  (ed_status.satisfied AND ed_status.count=2) OR
    scaleptflag DO
    BEGIN
        ed_iden;scaleptflag:=FALSE;
        psx:=intty.r1;psy:=intty.r2;
        px:=intty.r3;py:=intty.r4;
        translate(-px,-py);scale(psx,psy);
        translate(px,py);
        IF edit THEN interact;
```

```
    END;
    master:=FALSE;
END;
PROCEDURE reflect_x;
BEGIN
    opcode:=31;IF Open THEN transfer(0,0,0,0,0,0) ELSE
    IF edit THEN BEGIN
        ed_temp(2,2):=-1;ed_temp(1,1):=1;
        ed_temp(3,3):=1;ed_temp(1,2):=
        ed_temp(1,3):=ed_temp(2,1):=ed_temp(2,3)
        :=ed_temp(3,1):=ed_temp(3,2):=0.0;
        ed_trfmul;
    END
    ELSE
    BEGIN
        int_temp;temp(2,2):=-1;trf_mul;
    END;
END;
PROCEDURE reflect_y;
BEGIN
    opcode:=32;IF Open THEN transfer(0,0,0,0,0,0) ELSE IF
    edit THEN
    BEGIN
        ed_temp(1,1):=-1;
        ed_temp(2,2):=1;ed_temp(3,3):=1;ed_temp(1,2):=
        ed_temp(1,3):=ed_temp(2,1):=ed_temp(2,3):=
        ed_temp(3,1):=ed_temp(3,2):=0.0;
        ed_trfmul;
    END
    ELSE
    BEGIN
        int_temp;temp(1,1):=-1;trf_mul;
    END;
END;
PROCEDURE reflect_xoy;
BEGIN
    opcode:=33;IF Open THEN transfer(0,0,0,0,0,0) ELSE IF
    edit THEN
    BEGIN
        ed_temp(1,1):=-1;
        ed_temp(2,2):=-1;ed_temp(3,3):=1;ed_temp(1,2):=
        ed_temp(1,3):=ed_temp(2,1):=ed_temp(2,3):=
        ed_temp(3,1):=ed_temp(3,2):=0.0;
        ed_trfmul;
    END     ELSE
    BEGIN
        int_temp;temp(1,1):=temp(2,2):=-1;trf_mul;
    END;
END;
PROCEDURE get_trfmatrix(n):INTEGER n;
BEGIN
```

```
    INTEGER i,j;
    j:=pict_counter-1;
    FOR i:=1 STEP 1 UNTIL 1  DO
    BEGIN
        IF pict_table(i,1)=n THEN
        BEGIN
            temp(1,1):=pict_table(i,2);
            temp(1,2):=pict_table(i,3);
            temp(1,3):=pict_table(i,4);
            temp(2,1):=pict_table(i,5);
            temp(2,2):=pict_table(i,6);
            temp(2,3):=pict_table(i,7);
            temp(3,1):=pict_table(i,8);
            temp(3,2):=pict_table(i,9);
            temp(3,3):=pict_table(i,10);
            trf_mul;
            GOTO return;
        END
    END;
return:END;
PROCEDURE post(n);INTEGER n;
BEGIN
    REAL ARRAY data(1:6);
    INTEGER i,j,op,ka;
    IF Open THEN BEGIN
        Outimage;
        Outtext(" GSIM   ERROR #4.1;;");
        Outimage;GO TO done;
    END
    ELSE  BEGIN
        post_flag:=TRUE;
        ka:=prgm_counter-1;
        FOR i:=1 STEP 1 UNTIL ka  DO
        BEGIN
            IF seg_opcode_table(i,1)=n THEN
            BEGIN
                op:=seg_opcode_table(i,2);
                FOR j:=1 STEP 1 UNTIL 6 DO
                data(j):=attributes(i,j);
                get_trfmatrix(n);
                shiftx:=0;shifty:=0;
                IF op=10 THEN
                setwindow(data(1),data(2),
                data(3),data(4))ELSE
                IF op=11 THEN
                setviewport(data(1),data(2),
                data(3),data(4))ELSE

                IF op=9 THEN moveto(data(1),data(2)) ELSE
                IF op=12 THEN drawto(data(1),data(2));
                IF op=0 THEN  exit  ELSE
```

```
                IF op=1 THEN   newpage ELSE
                IF op=2 THEN bell;
            END;
        END;
    END;
    post_flag:=  FALSE;
    iden;
END;

PROCEDURE pri_pict_table;
BEGIN
    INTEGER i,j;
    FOR i:=1 STEP 1 UNTIL  5  DO BEGIN
        FOR j:=1 STEP 1 UNTIL 10 DO
        Outreal(pict_table(i,j),10,7);
        Outimage;
    END;
END;
PROCEDURE pri_seq_opcode_table;
BEGIN
    INTEGER i,j;
    FOR i:=1 STEP 1UNTIL 10 DO
    FOR j:=1 STEP 1 UNTIL  2 DO
    BEGIN
    Outint(seq_opcode_table(i,j),2);END;
    Outimage;
END;
PROCEDURE  save_wvconstant;
BEGIN
    s1:=wvconstant.wvxm;s5:=wxlc;
    s2:=wvconstant.wvxa;s6:=wxrc;
    s3:=wvconstant.wvym;s7:=wylc;
    s4:=wvconstant.wvya;s8:=wyrc;
END;

PROCEDURE getback;
BEGIN
    wvconstant.wvxm:=s1;wxlc:=s5;
    wvconstant.wvxa:=s2;wxrc:=s6;
    wvconstant.wvym:=s3;wylc:=s7;
    wvconstant.wvya:=s4;wyrc:=s8;
END;
PROCEDURE setwvport_ed;
BEGIN
    save_wvconst;
    setwindow(0,10,0,10);
    setviewport(780,1020,0,780);
END;
PROCEDURE increment;
BEGIN
    xinc:=((window.wxr-window.wxl)/780)*10;
```

```
    vinc:=((window.wyt-window.wyb)/780)*10;
END;
PROCEDURE cirpanel(px,py,pp,pnl,pangle);REAL px,py,pp;
INTEGER pnl ,pangle;
BEGIN
    REAL tempx,tempy,cy,cx,delv,disc,ang,x,y,p;
    INTEGER i,nl,angle;BOOLEAN cirpanflag;
    opcode:=34;complex:=TRUE;
    intty.r1:=x:=px;intty.r2:=y:=py;intty.r3:=p:=pp;
    intty.i1:=nl:=pnl;intty.i2:=angle:=pangle;
    log_error:=TRUE;
    WHILE log_error DO IF
    ((intty.r3 <= 0 ) OR (intty.i1 <= 0)
    OR (intty.i2 <=
    0))
    THEN
    interact ELSE log_error:=
    FALSE;ints_status; cirpanflag:=TRUE;
    WHILE NOT(ed_status.satisfied OR ed_status.count=2) OR
    cirpanflag DO BEGIN
        cirpanflag:=FALSE;
        x:=intty.r1;y:=intty.r2;p:=intty.r3;
        nl:=intty.i1;angle:=intty.i2;
        delv:=(2*p)/nl;ang:=angle*d_to_r;
        cx:=tempx:=x*Cos(ang)+y*Sin(ang);
        cy:=tempy:=x*(-Sin(ang))+y*Cos(ang);
        tempy:=tempy-p;
        WHILE tempy < (cy+p-delv)   DO
        BEGIN
            tempy:=tempy+delv;
            disc:=Sqrt(p^2-(tempy-cy)^2);
            moveto(((cx+disc)*Cos(ang)-
            tempy*Sin(ang)),
            (cx+disc)*Sin(ang)+tempy*Cos(ang));
            drawto(((cx-disc)*Cos(ang)-tempy*Sin(ang)),
            (cx-disc)*Sin(ang)+tempy*Cos(ang));
        END;
        intty.r1:=x;intty.r2:=y;intty.r3:=
        p;intty.i1:=nl;intty.i2:=angle;
    IF edit THEN        interact;END;
    complex:=FALSE;
END;
PROCEDURE polypanel(n,vx,vv,nl,sangle);
REAL ARRAY  vx,vv;INTEGER  n,nl,sangle;
BEGIN
    INTEGER direct,angle;
    REAL  minx,maxx,miny,maxy;
    BOOLEAN  arcflag,flag;
    INTEGER ptcount;REAL ARRAY ptsx(1:n),ptsv(1:n);
    PROCEDURE  box;
    BEGIN
```

```
    REAL  temp,ang;
    INTEGER  i;
    ang:=angle*d_to_r;
    minx:=maxx:=vx(1)*Cos(ang)-direct*vy(1)*Sin(ang);
    miny:=maxy:=direct*vx(1)*Sin(ang)+vy(1)*Cos(ang);
    FOR i:=1 STEP 1 UNTIL n DO
    BEGIN
        temp:=vx(i)*Cos(ang)-direct*vy(i)*Sin(ang);
        IF temp >  maxx  THEN  maxx:=temp;
        IF  temp < minx  THEN  minx:=temp;
        temp:=direct*vx(i)*Sin(ang)+vy(i)*Cos(ang);
        IF  temp > maxy  THEN  maxy:=temp;
        IF temp < miny  THEN  miny:=temp;
    END;
END;
PROCEDURE sort_draw;
BEGIN
    INTEGER i,j;REAL temp;
    FOR i:=1 STEP 1 UNTIL ptcount DO
    FOR j:=i STEP 1 UNTIL ptcount DO
    BEGIN
        IF ptsx(j)  <  ptsx(i)  THEN
        BEGIN
            temp:=ptsx(i);ptsx(i):=ptsx(j);
            ptsx(j):=temp;
            temp:=ptsy(i);ptsy(i):=ptsy(j);
            ptsy(j):=temp;
        END;
    END;
    i:=1;
    WHILE   i<  ptcount  DO
    BEGIN
        moveto(ptsx(i),ptsy(i));
        drawto(ptsx(i+1),ptsy(i+1));
        i:=i+2;
    END;
    ptcount:=0;
END;
PROCEDURE drawpanel;
BEGIN
    REAL  x,y,dely,tempy1,tempy2,tempx1,tempx2,ang;
    INTEGER i,i1;
    flag:=FALSE;ang:=angle*d_to_r;
    box; dely:=(maxy-miny)/n1;
    y:=miny;ptcount:=0;
    WHILE  y <   (maxy-dely)  DO
    BEGIN
        y:=y+dely;
        FOR i:=1 STEP 1 UNTIL  n  DO
        BEGIN
            IF i=n  THEN  i1:=1 ELSE i1:=i+1;
```

```
                tempy1:=direct*vx(i)*Sin(ang)
                +vv(i)*Cos(ang);
                tempy2:=direct*vx(il)*
                Sin(ang)+vv(il)*Cos(ang);
                IF tempy1  \=  tempy2  THEN
                BEGIN
                    tempx1:=vx(i)*
                    Cos(ang)-direct*vy(i)*Sin(ang);
                    tempx2:=vx(il)*
                    Cos(ang)-direct*vy(il)*Sin(ang);
                    x:=(((y-tempy1)*
                    (tempx2-tempx1))/
                    (tempy2-tempy1))+
                    tempx1;arcflag:=FALSE;
                    IF NOT master THEN BEGIN
                        IF ((x >  tempx1 AND x <= tempx2)  OR
                        (x < tempx1  AND x > tempx2))  THEN
                        arcflag:=TRUE;
                    END ELSE
                    BEGIN
                        IF ((x>=tempx1 AND x <= tempx2 ) OR
                        (x <=tempx1 AND x >=
                        tempx2)) THEN
                    arcflag:=TRUE;END;
                    IF arcflag THEN BEGIN
                        ptcount:=ptcount+1;
                        ptsx(ptcount):=x*Cos(ang)+direct*y*Sin(ang);
                        ptsy(ptcount):=-x*direct*Sin(ang)+y*Cos(ang);
                    END;
                END;
            END;
            IF ptcount > 0 THEN sort_draw;
        END;
    END;
    IF NOT master THEN BEGIN opcode:=35;complex:=TRUE;END;
    IF ((n<=0) OR (n1<=0) OR (sangle<0) OR (sangle>180))
    THEN BEGIN
    Outtext(" GSIM ERROR  #  ");Outimage;END
    ELSE  BEGIN
        direct:=1;
        IF sangle <= 90 THEN angle:=-sangle ELSE
        IF sangle <= 180 THEN angle:=-(sangle-180);
        drawpanel;
    END;
    IF NOT master THEN      complex:=FALSE;
END;
PROCEDURE findangle(xc,yc,xp,yp);
REAL xc,yc,xp,yp;
BEGIN
    REAL   tempang;
    IF  ((xp>xc) AND (yp>yc)) THEN tempang:=
```

```
    Arctan((yp-yc)/(xp-xc))*r_to_d ELSE
    IF  ((xp<xc) AND (yp>yc)) THEN
    tempang:=180-((Arctan((yp-yc)/(xc-xp)))*r_to_d) ELSE
    IF  ((xp<xc) AND (yp<yc)) THEN
    tempang:=180+((Arctan((yc-yp)/(xc-xp)))*r_to_d) ELSE
    IF  ((xp>xc) AND (yp<yc)) THEN
    tempang:=360-((Arctan((yc-yp)/(xp-xc)))*r_to_d) ELSE
    IF ((xp=xc) AND (yp>yc)) THEN tempang:=90 ELSE
    IF ((yp=yc) AND (xp<xc)) THEN tempang:=180 ELSE
    IF ((xp=xc) AND (yp<yc)) THEN tempang:=270 ELSE
    IF ((yp=yc) AND (xp>xc)) THEN tempang:=0;
    arc_angle:=tempang;
END;
PROCEDURE arc3pt(p,q,r,s,t,u);REAL p,q,r,s,t,u;
BEGIN
    REAL x1,y1,x2,y2,m1,m2,c1,c2,radius,a1,a2,x,y;

    opcode:=36;complex:=TRUE;master:=TRUE;
    IF Sqrt((p-t)^2+(q-u)^2)=(Sqrt((p-r)^2+(q-s)^2)
    +Sqrt((r-t)^2+(s-u)^2))
    THEN
    BEGIN
    moveto(p,q);drawto(t,u);END ELSE
    BEGIN
        moveto(p,q);x1:=(p+r)/2;y1:=(q+s)/2;x2:=(r+t)/2;y2:=(s+u)/2;
        m1:=-((r-p)/(s-q));m2:=-((t-r)/(u-s));
        c1:=y1-m1*x1;c2:=y2-m2*x2;x:=(c2-c1)/(m1-m2);
        y:=m1*x+c1;
        findangle(x,y,p,q);a1:=arc_angle;
        findangle(x,y,t,u);a2:=arc_angle;
        radius:=Sqrt((q-y)^2+(p-x)^2);
        arc(radius,a1,a2);
    END;
    complex:=FALSE;master:=FALSE;
END;
PROCEDURE arcpanel(p,stangle,endangle,n1,slope);
REAL  p;INTEGER stangle,endangle,n1,slope;
BEGIN
    INTEGER i,j,k;REAL ARRAY vx(1:180),vy(1:180);
    opcode:=37;complex:=TRUE;master:=TRUE;
    vx(2):=(olddvst.cx-wvconstant.wvxa)/wvconstant.wvxm;
    vy(2):=(olddvst.cy-wvconstant.wvya)/wvconstant.wvym;
    vx(1):=vx(2)-p*Cos(stangle*d_to_r);
    vy(1):=vy(2)-p*Sin(stangle*d_to_r);
    j:=0;k:=(endangle-stangle)//2;
    FOR i:=1 STEP 1 UNTIL k DO
    BEGIN
        vx(i+2):=vx(1)+p*Cos((stangle+2*i)*d_to_r);
        vy(i+2):=vy(1)+p*Sin((stangle+2*i)*d_to_r);
        j:=j+1;
    END;
```

```
    polypanel(j+2,vx,vy,nl,slope);
    master:=FALSE;complex:=FALSE;
END;
PROCEDURE cir3pt(pp,pq,pr,ps,pt,pu);REAL pp,pq,pr,ps,pt,pu;
BEGIN
    REAL  l1,l2,l3,x1,x2,y1,y2,x,y,radius,c1,c2,m1,m2;
    REAL p,q,r,s,t,u;
    intty.r1:=p:=pp;intty.r2:=q:=pq;intty.r3:=r:=pr;
    intty.r4:=s:=ps;intty.r5:=t:=pt;intty.r6:=u:=pu;
    cir3ptflag:=TRUE;opcode:=38;master:=TRUE;complex:=TRUE;
    l1:=Sqrt((p-t)^2+(q-u)^2);
    l2:=Sqrt((p-r)^2+(q-s)^2);
    l3:=Sqrt((r-t)^2+(s-u)^2);
    IF ((l1+l2=l3) OR (l2+l3=l1)  OR (l3+l1=l2)) THEN
    BEGIN
        moveto(p,q);drawto(t,u);
    END ELSE
    BEGIN
        ints_status;
        WHILE NOT(ed_status.satisfied AND ed_status.count=2) OR
        cir3ptflag DO BEGIN
            p:=intty.r1;q:=intty.r2;r:=intty.r3;
            s:=intty.r4;t:=intty.r5;u:=intty.r6;
            master:=TRUE;complex:=TRUE;cir3ptflag:=FALSE;
            x1:=(p+r)/2;y1:=(q+s)/2;x2:=(r+t)/2;y2:=(s+u)/2;
            m1:=-((r-p)/(s-q));m2:=-((t-r)/(u-s));
            c1:=y1-m1*x1;c2:=y2-m2*x2;x:=(c2-c1)/(m1-m2);y:=m1*x+c1;
            radius:=Sqrt((x-p)^2+(y-q)^2);circle(x,y,radius);
            intty.r1:=p;intty.r2:=q;intty.r3:=r;intty.r4:=s;
            intty.r5:=t;intty.r6:=u;
        IF edit THEN              interact;END;
    END;
    complex:=FALSE;master:=FALSE;cir3ptflag:=FALSE;
END;

PROCEDURE chdrawto(x,y,n);REAL x,y;INTEGER n;
BEGIN
    REAL p,q,sint,cost,Length,x1,y1,x2,y2,i1,i2,i3,i4,i5,i6,i7,k1,k2;
    REAL k3,i8,i9,i10,i11,i12,u;
    BOOLEAN   stline,flag1,flag2;
    INTEGER jcou,count;
    PROCEDURE findlimit;
    BEGIN
        REAL limit;
        IF n=0 THEN limit:=10.5*u ELSE
        IF n=1 THEN limit:=8*u ELSE
        IF n=2 THEN limit:=12*u ELSE
        IF n=3 THEN limit:=11*u  ELSE
        IF n=4 THEN limit:=9.6*u ELSE
        IF n=5 THEN limit:=5.6*u;
        IF flag1 THEN BEGIN
```

```
        IF (x1-limit/2) >= x2 THEN GO TO retlt
    END ELSE
    BEGIN
        IF (y1-limit/2) >=y2 THEN GO TO retlt;
    END
END;
flag1:=FALSE;flag2:=FALSE;opcode:=39;u:=(10/780);
stline:=TRUE;
p:=(olddvst.cx-wvconstant.wvxa)/wvconstant.wvxm;
q:=(olddvst.cy-wvconstant.wvya)/wvconstant.wvym;
complex:=TRUE;
IF (q=y)  THEN BEGIN sint:=0;cost:=1;
    flag1:=TRUE;IF  p<x  THEN BEGIN  x1:=p;
        Length:=x-p;
    y1:=y2:=y;x2:=x;END
    ELSE BEGIN y1:=y2:=y;x1:=x;x2:=p;Length:=p-x;END
END
ELSE IF (p=x)  THEN  BEGIN  flag2:=TRUE;sint:=1;cost:=0;
    IF  q<y THEN BEGIN   x1:=x2:=p;y1:=q;
        Length:=y-q;
    y2:=y;END
    ELSE BEGIN.
        Length:=q-y;
x1:=x2:=p;y1:=y;y2:=q END END
ELSE BEGIN
    IF  q<y THEN. BEGIN
    x1:=p;y1:=q;x2:=x;y2:=y;END  ELSE
    BEGIN
    x1:=x;y1:=y;x2:=p;y2:=q;END;
    Length:=Sqrt((x-p)^2+(y-q)^2);
    cost:=(x2-x1)/Length;sint:=(y2-y1)/Length;
END;
k1:=(wxrc-wxlc);k2:=(wyrc-wylc);k3:=(20/780);
IF n=0 THEN BEGIN  i1:=6*u;i2:=1.5*u;i3:=1.5*u;i4:=1.5*u;
i5:=i6:=i7:=i8:=i9:=i10:=i11:=i12:=0;END
ELSE
IF n=1 THEN   BEGIN
    i1:=i3:=i5:=i7:=2*u;
    i2:=i4:=i6:=i8:=u;i9:=i10:=i11:=i12:=0;
END ELSE
IF n=2 THEN BEGIN
    i1:=i3:=i5:=3*u;
    i2:=i4:=i6:=u;
    i7:=i8:=i9:=i10:=i11:=i12:=0;
END ELSE
IF n=3 THEN   BEGIN
    i1:=i9:=2*u;
    i2:=i3:=i4:=i5:=i6:=i7:=i8:=u;
    i10:=u;i11:=i12:=0;
END ELSE
IF n=4 THEN BEGIN
```

```
        i1:=i2:=i3:=i4:=i5:=i6:=i7:=i8:=i9:=
    i10:=i11:=i12:=.8*u; END
    ELSE
    IF n=5 THEN BEGIN
        i1:=i3:=i5:=i7:=i9:=i11:=.1*u;
        i2:=i4:=i6:=i8:=i10:=1*u;
    END ELSE BEGIN moveto(x1,y1);drawto(x2,y2); stline:=FALSE;END;
    IF n<4 THEN count:=(Length/(12*u))//1 ELSE;
    IF n=4 THEN count:=(Length/(3.6*u))//1 ELSE
    IF n=5 THEN count:=(Length/(7*u))//1 ELSE
    IF n=6 THEN count:=(Length/(5*u))//1;
    IF stline THEN BEGIN
        cdrw:
        BEGIN
            moveto(x1,y1);x1:=x1+i1*k1*cost;
            y1:=y1+i1*k2*sint;findlimit;
            drawto(x1,y1);x1:=x1+i2*k1*cost;
            y1:=y1+i2*k2*sint;findlimit;
            moveto(x1,y1);x1:=x1+i3*k1*cost;
            y1:=y1+i3*k2*sint;findlimit;
            drawto(x1,y1);x1:=x1+i4*k1*cost;
            y1:=y1+i4*k2*sint;findlimit;
            moveto(x1,y1);x1:=x1+i5*k1*cost;
            y1:=y1+i5*k2*sint;findlimit;
            drawto(x1,y1);x1:=x1+i6*k1*cost;
            y1:=y1+i6*k2*sint;findlimit;
            moveto(x1,y1);x1:=x1+i7*k1*cost;
            y1:=y1+i7*k2*sint;findlimit;
            drawto(x1,y1);x1:=x1+i8*k1*cost;
            y1:=y1+i8*k2*sint;findlimit;
            moveto(x1,y1);x1:=x1+i9*k1*cost;
            y1:=y1+i9*k2*sint;findlimit;
            drawto(x1,y1);x1:=x1+i10*k1*cost;
            y1:=y1+i10*k2*sint;findlimit;
            moveto(x1,y1);x1:=x1+i11*k1*cost;
            y1:=y1+i11*k2*sint;findlimit;
            drawto(x1,y1);x1:=x1+i12*k1*cost;
            y1:=y1+i12*k2*sint;findlimit;
            IF flag1 THEN BEGIN IF x1<x2 THEN GO TO cdrw END ELSE
            BEGIN IF y1<y2 THEN GO TO cdrw;END;
        END;
        drawto(x,y);
        retlt:    complex:=FALSE;
    END;
END;
PROCEDURE regpol(pxp,pyp,pradius,pnumsid);REAL pxp,pyp,pradius;
INTEGER  pnumsid;
BEGIN
    REAL  x,y,ang,xp,yp,radius;
    INTEGER i,numsid;BOOLEAN regpolflag;
    complex:=TRUE;regpolflag:=TRUE;opcode:=40;
```

```
    intty.r1:=xp:=pxp;intty.r2:=yp:=pyp;intty.r3:=radius:=pradius;
    intty.i1:=numsid:=pnumsid;
    log_error:=TRUE;
    WHILE log_error DO IF (intty.r3 <= 0) OR (intty.i1 <= 0) THEN
    interact ELSE log_error:=FALSE;
    ints_status;regpolflag:=TRUE;
    WHILE NOT ((ed_status.satisfied AND ed_status.count=2)
    OR master)  OR regpolflag  DO
    BEGIN       regpolflag:=FALSE;complex:=TRUE;

        ang:=(360/numsid);
        moveto(xp+radius,yp);
        FOR i:=1 STEP 1 UNTIL numsid DO
        BEGIN
            x:=xp+radius*Cos(ang*i*d_to_r);
            y:=yp+radius*Sin(ang*i*d_to_r);
            drawto(x,y);
        END;
        intty.r1:=xp;intty.r2:=yp;
        intty.r3:=radius;intty.i1:=numsid;

    interact;END;
    IF NOT master THEN      complex:=FALSE;
END;

PROCEDURE  regpolpanel(pxp,pyp,pradius,pnumsid,pnl,pslope);
REAL pxp,pyp,pradius;
INTEGER pnumsid,pnl,pslope;
BEGIN
    REAL ARRAY ax(1:pnumsid),ay(1:pnumsid);REAL  ang;
    REAL xp,yp,radius;        INTEGER numsid,nl,slope;
    INTEGER sides, i;BOOLEAN regpolpanflag;
    intty.r1:=xp:=pxp;
    intty.r2:=yp:=pyp;intty.r3:=radius:=pradius;
    intty.i1:=numsid:=pnumsid;intty.i2:=nl:=pnl;
    intty.i3:=slope:=pslope;regpolpanflag:=TRUE;opcode:=41;
    log_error:=TRUE;
    WHILE log_error DO IF (intty.r3 <= 0) OR
    (intty.i1 <= 0)  OR (intty.i2 <= 0) OR
    (intty.i3 <= 0 OR intty.i3 >= 180
    )       THEN  interact ELSE log_error:=FALSE;   ints_status;
    WHILE NOT (ed_status.satisfied AND ed_status.count=2)
    OR regpolpanflag DO BEGIN
        regpolpanflag:=FALSE;master:=TRUE;complex:=TRUE;
        xp:=intty.r1;yp:=intty.r2;radius:=intty.r3;
        numsid:=intty.i1;nl:=intty.i2;slope:=intty.i3;
        sides:=numsid-1;ang:=(360/numsid);
        ax(1):=xp+radius;ay(1):=yp;
        FOR i:=1 STEP 1 UNTIL sides DO
        BEGIN
            ax(i+1):=xp+radius*Cos(ang*i*d_to_r);
```

```
            ay[i+1]:=yp+radius*sin(ang*i*d_to_r);
        END;
        polypanel(numsid,ax,ay,nl,slope);
        interact;
    END;
    master:=FALSE;complex:=FALSE;
END;
PROCEDURE edit_pict(n);INTEGER n;
BEGIN
    IF (Open) THEN BEGIN
    Outtext(  "GSIM  ERROR #3  ");Outimage;GO TO done; END ELSE
    BEGIN
        IF edit THEN BEGIN     Outtext("GSIM ERROR #2 ");
        Outimage;GO TO done;END ELSE BEGIN
            c_ed_prgmcounter:=0;edit:=TRUE;
            ed_pictno:=n; iden;ed_iden;
        END;
    END;
END;
PROCEDURE  end_pict(n);INTEGER n;
BEGIN
    IF    (NOT  edit) THEN
    BEGIN
    Outtext("GSIM    ERROR    #5");Outimage;GO TO done; END ELSE
    BEGIN
    edit:=FALSE;  iden; END;
END;

PROCEDURE end_save_pict(n);INTEGER n;
BEGIN
    INTEGER  i;
    IF ((Open)  OR (NOT edit) OR (ed_pictno \= n)) THEN
    BEGIN
    Outtext("   GSIM ERROR #3  ");GO TO done;Outimage;  END  ELSE
    BEGIN
        edit:=FALSE; FOR i:=1 STEP 1 UNTIL c_ed_prgmcounter DO
        BEGIN
            ed_no_opcode(ed_prgmcounter+i,1):=temp_ed_no_opcode(i,1);
            ed_no_opcode(ed_prgmcounter+i,2):=temp_ed_no_opcode(i,2);
            ed_attr(ed_prgmcounter+i,1):=temp_ed_attr(i,1);
            ed_attr(ed_prgmcounter+i,2):=temp_ed_attr(i,2);
        END;
    ed_prgmcounter:=ed_prgmcounter+c_ed_prgmcounter;END;
END;

PROCEDURE editedxy_save(x,y);REAL x,y;
BEGIN
    c_ed_prgmcounter:=c_ed_prgmcounter+1;
    temp_ed_no_opcode(c_ed_prgmcounter,1):=ed_pictno;
    temp_ed_no_opcode(c_ed_prgmcounter,2):=opcode;
    temp_ed_attr(c_ed_prgmcounter,1):=x;
```

```
    temp_ed_attr(c_ed_prgmcounter,2):=y;
END;
PROCEDURE replay_edited(n);INTEGER n;
BEGIN
    INTEGER i;opcode:=42;
    IF  ed_prgmcounter = 0 THEN BEGIN  END ELSE
    BEGIN
        FOR i:=1 STEP 1 UNTIL ed_prgmcounter DO
        BEGIN
            IF ed_no_opcode(i,1)=n THEN
            BEGIN
                IF ed_no_opcode(i,2)=6 THEN
                ed_moveto(ed_attr(i,1),ed_attr(i,2))  ELSE
                ed_drawto(ed_attr(i,1),ed_attr(i,2));
            END;
        END;
    END;
END;

PROCEDURE ed_moveto(x,y);REAL  x,y;
BEGIN
    olddvst.cx:=     edmx:=wvconstant.wvxm*x+wvconstant.wvxa;
    olddvst.cy:=     edmy:=wvconstant.wvym*y+wvconstant.wvya;
END;

PROCEDURE  ed_drawto(x,y);REAL x,y;
BEGIN
    REAL nx,ny;
    nx:=wvconstant.wvxm*x+wvconstant.wvxa;
    ny:=wvconstant.wvym*y+wvconstant.wvya;
    clip(edmx,edmy,nx,ny);olddvst.cx:=
    edmx:=nx;olddvst.cy:=edmy:=ny;

END;
PROCEDURE break;
BEGIN
    TEXT tw;
    opcode:=43;tw:=Copy("CR");
    save_wvconstants;
    setwindow(0,100,0,100);
    setviewport(-240,0,0,240);
    savesize;
    bringtext(0,50,tcr);bringtext(50,50,tw);
    getsize;
    Outimage;Inimage;
    getback;
END;
PROCEDURE savesize;
BEGIN
    tempsizex:=sca_d_x;tempsizey:=sca_ch_y;
    scale_char(.5,.5);
```

```
END;

PROCEDURE getsize;
BEGIN
    sca_d_x:=tempsizex;
    sca_d_y:=tempsizey;
END;
PROCEDURE  interact;
BEGIN
    BOOLEAN cont_error,stline;
    PROCEDURE st_line;
    BEGIN
        REAL p,q,r,s,t,u,l1,l2,l3;
        p:=intty.r1;q:=intty.r2;r:=intty.r3;s:=intty.r4;
        t:=intty.r5;u:=intty.r6;
        l1:=Sqrt((p-t)^2+(q-u)^2);
        l2:=Sqrt((p-r)^2+(q-s)^2);
        l3:=Sqrt((r-t)^2+(s-u)^2);
        IF ((l1+l2=l3) OR (l2+l3=l1) OR (l3+l1=l2))
        THEN stline:=TRUE        ELSE stline:=FALSE;
    END;
    IF ed_status.count \= 1 OR log_error  THEN
    BEGIN
        IF edit OR log_error  THEN
        BEGIN
            IF log_error THEN GOTO detect;
            Outtext("Y TO MODIFY ELSE N");Outimage;Inimage;
            ;ed_status.ch:=Inchar;IF ed_status.ch ='Y' THEN
            BEGIN
                re_draw_edited;
                detect:   IF opcode=3 THEN
                BEGIN
                    Outtext("  TRANSLATE");Outimage;
                    Outfix(intty.r1,2,9);Outfix(intty.r2,2,9);

                    Outimage;
                    Outimage;Outtext("GIVE NEW VALUES");Outimage;
                    intty.r1:=Inreal;intty.r2:=Inreal;
                    ed_temp(1,1):=ed_temp(2,2):=ed_temp(3,3):=1.0;
                    ed_temp(3,1):=intty.r1;ed_temp(3,2):=intty.r2;
                    ed_temp(1,2):=ed_temp(1,3):=
                    ed_temp(2,1):=ed_temp(2,3):=0.0;

                    ed_trfmul;newpage;
                    re_draw_edited;
                END ELSE
                IF opcode=5 THEN
                BEGIN
                    Outtext("ROTATEPT");Outimage;
                    Outfix(intty.r1,2,9);Outfix(intty.r2,2,9);
                    Outfix(intty.r3,2,9);
```

```
    Outtext("GIVE NEW VALUES");Outimage;intty.r1:=Inreal;
    intty.r2:=Inreal;intty.r3:=Inreal;
    ed_temp(1,1):=ed_temp(2,2):=Cos(intty.r1*d_to_r);
    ed_temp(2,1):=Sin(intty.r1*d_to_r);
    ed_temp(1,2):=-ed_temp(2,1);
    ed_temp(3,1):=-Cos(intty.r1*d_to_r)*intty.r2-
    Sin(intty.r1*d_to_r)*intty.r3;
    ed_temp(3,2):=Sin(intty.r1*d_to_r)*intty.r2-
    intty.r3*Cos(intty.r1*d_to_r);
    ed_temp(3,3):=1.0;ed_temp(1,3):=
    ed_temp(2,3):=0;ed_trfmul;
    newpage;re_draw_edited;
END ELSE
IF opcode=9 THEN
BEGIN
    Outtext("MOVETO");Outimage;
    Outfix(intty.r1,2,9);Outfix(intty.r2,2,9);
    Outimage;
    Outtext("GIVE NEW VALUES");Outimage;
    intty.r1:=Inreal;intty.r2:=Inreal;newpage;
re_draw_edited;END ELSE
IF opcode=12 THEN
BEGIN
    Outtext("DRAWTO");Outimage;
    Outfix(intty.r1,2,9);Outfix(intty.r2,2,9);Outimage;
    Outtext("GIVE NEW VALUES");Outimage;
    intty.r1:=Inreal;intty.r2:=Inreal;newpage;
    re_draw_edited;
END ELSE


IF opcode=10 THEN
BEGIN

    IF log_error THEN
    BEGIN
        cont_error:=TRUE;
        WHILE cont_error DO
        BEGIN
            IF ((intty.r1 >= intty.r2) OR
            (intty.r3 >= intty.r4)) THEN
            cont_error:=TRUE ELSE cont_error:=FALSE;
            IF cont_error THEN
            BEGIN
                Outtext
                ("ERROR  IN DATA  SETWINDOW  ");Outimage;
                Outfix(intty.r1,2,9);Outfix(intty.r2,2,9);
                Outfix(intty.r3,2,9);Outfix(intty.r4,2,9);
                Outimage;
```

```
                    Outtext(" GIVE THE CORRECT VALUES");
                    Outimage;
                    intty.r1:=Inreal;intty.r2:=Inreal;
                    intty.r3:=Inreal;intty.r4:=Inreal;
                    ;
                END;
            END;
        END  ELSE
        BEGIN
            Outtext("  SETWINDOW");Outimage;Outimage;
            Outfix(intty.r1,2,9);Outfix(intty.r2,2,9);
            Outfix(intty.r3,2,9);Outfix(intty.r4,2,9);
            Outimage;
            Outtext("GIVE NEW VALUES"); Outimage;
            intty.r1:=Inreal;intty.r2:=Inreal;
            intty.r3:=Inreal;intty.r4:=Inreal;newpage;
            window.wxl:=intty.r1;window.wxr:=intty.r2;
            window.wyb:=intty.r3;window.wyt:=intty.r4;
            wxrc:=intty.r2;wxlc:=intty.r1;
            wyrc:=intty.r4;wylc:=intty.r3;
            set_w_v_constants;
            re_draw_edited;
        END;
END
ELSE                        IF opcode=4 THEN
BEGIN
        Outtext("ROTATED");Outimage;Outimage;
        Outfix(intty.r1,2,9);
        Outimage;
        Outtext("GIVE NEW VALUE");Outimage;
        intty.r1:=Inreal;
        ed_temp(1,3):=ed_temp(2,3):=
        ed_temp(3,1):=ed_temp(3,2):=0.0;
        ed_temp(3,3):=1.0;ed_temp(1,1):=
        ed_temp(2,2):=Cos(intty.r1*d_to_r);
        ed_temp(2,1):=-Sin(intty.r1*d_to_r);
        ed_temp(1,2):=-ed_temp(2,1);
        ed_trfmul;
        newpage;re_draw_edited;
END ELSE


IF opcode=11 THEN
BEGIN
        IF log_error THEN
        BEGIN
            cont_error:=TRUE;
            WHILE cont_error DO
            BEGIN
                IF ((intty.r1 >=intty.r2) OR (intty.r3
```

```
                >= intty.r4))  THEN  cont_error :=

                TRUE ELSE
                cont_error:=FALSE;
                IF cont_error THEN
                BEGIN
                    Outtext
                    ("ERROR IN DATA SETVIEWPORT  ");Outimage;
                    Outfix(intty.r1,2,9);Outfix(intty.r2,2,9);
                    Outfix(intty.r3,2,9);Outfix(intty.r4,2,9);
                    Outimage;
                    Outtext("GIVE CORRECT VALUES");Outimage;
                    Outimage;
                    intty.r1:=Inreal;intty.r2:=Inreal;
                    intty.r3:=Inreal;intty.r4:=Inreal;
                END;
            END;

        END ELSE

        BEGIN
            Outtext("SETVIEWPORT");Outimage;Outimage;
            Outfix(intty.r1,2,9);Outfix(intty.r2,2,9);
            Outfix(intty.r3,2,9);Outfix(intty.r4,2,9);
            Outimage;
            Outtext("GIVE NEW VALUES");Outimage;
            intty.r1:=Inreal;intty.r2:=Inreal;
            intty.r3:=Inreal;intty.r4:=Inreal;
            clipconstant.clipxl:=viewport.vxl:=intty.r1;
            clipconstant.clipxr:=viewport.vxr:=intty.r2;
            clipconstant.clipyb:=viewport.vyb:=intty.r3;
            clipconstant.clipyt:=viewport.vyt:=intty.r4;
            set_w_v_constants;
            newpage;re_draw_edited;
        END;
    END
    ELSE
    IF opcode=15 THEN
    BEGIN
        IF log_error THEN
        BEGIN
            cont_error:=TRUE;
            WHILE cont_error  DO
            BEGIN
                IF (intty.r3 <= 0 OR intty.r4 <= 0)
                THEN cont_error:=TRUE
                ELSE cont_error:=FALSE;
                IF cont_error  THEN BEGIN
                    Outtext("  ERROR IN DATA RECTANGLE");
                    Outfix(intty.r1,2,9);
                    Outfix(intty.r2,2,9);Outfix(intty.r3,2,9);
```

```
                    Outfix(intty.r4,2,9);Outimage;
                    Outtext("GIVE  CORRECT  VALUES");Outimage;
                    intty.r1:=Inreal;intty.r2:=Inreal;
                    intty.r3:=Inreal;intty.r4:=Inreal;

                    ;
                END;
            END;
        END
        ELSE
        BEGIN
            Outtext("RECTANGLE");Outimage;Outimage;
            Outfix(intty.r1,2,9);Outfix(intty.r2,2,9);
            Outfix(intty.r3,2,9);
            Outfix(intty.r4,2,9);
            Outimage;
            Outtext("GIVE NEW VALUES");Outimage;
            intty.r1:=Inreal;
            intty.r2:=Inreal;intty.r3:=Inreal;
            intty.r4:=Inreal;
            newpage;re_draw_edited;
        END;
    END ELSE

    IF opcode=14 THEN
    BEGIN
        Outint(opcode,3);
        IF log_error THEN
        BEGIN
            cont_error:=TRUE;
            WHILE cont_error DO
            BEGIN
                IF (intty.r3 <= 0) THEN cont_error
                :=TRUE ELSE cont_error:=FALSE;
                IF cont_error THEN
                BEGIN
                    Outtext
                    ("ERROR IN  DATA FOR CIRCLE  ");Outimage;
                    Outfix(intty.r1,2,9);
                    Outfix(intty.r2,2,9);Outfix(intty.r3,2,9);
                    Outimage;
                    Outtext
                    ("GIVE  CORRECT  VALUES");intty.r1:=Inreal;
                    intty.r2:=Inreal;intty.r3:=Inreal;
                END;
            END;
        END
        ELSE

        BEGIN
            Outimage;
```

```
            Outtext("CIRCLE");Outimage;Outimage;
            Outfix(intty.r1,2,9);Outfix(intty.r2,2,9);
            Outfix(intty.r3,2,9);
            Outimage;
            Outtext("GIVE NEW VALUES");
            Outimage;intty.r1:=Inreal;
            intty.r2:=Inreal;intty.r3:=Inreal;Outimage;
            newpage;re_draw_edited;
        END;
    END ELSE
    IF opcode=16  THEN
    BEGIN
        IF  log_error THEN
        BEGIN
            cont_error:=TRUE;
            WHILE cont_error DO
            BEGIN
                IF(intty.r3 <= 0.) THEN cont_error
                :=TRUE ELSE cont_error:=FALSE;
                IF cont_error THEN
                BEGIN
                    Outtext("ERROR  IN DATA  SQUARE");
                    Outimage;
                    Outfix(intty.r1,2,9);Outfix(intty.r2,2,9);
                    Outfix(intty.r3,2,9);

                    Outimage;
                    Outtext("GIVE  CORRECT  VALUES");
                    intty.r1:=Inreal;intty.r2:=Inreal;
                    intty.r3:=Inreal;
                END;
            END;
        END ELSE
        BEGIN
            Outtext("SQUARE");Outimage;Outimage;
            Outfix(intty.r1,2,9);Outfix(intty.r2,2,9);
            Outfix(intty.r3,2,9);Outimage;

            Outtext("GIVE NEW VALUES");Outimage;intty.r1:=Inreal;
            intty.r2:=Inreal;intty.r3:=Inreal;
            newpage;re_draw_edited;
        END;
    END ELSE
    IF opcode=21  THEN
    BEGIN
        Outtext("RMOVETO");Outimage;Outimage;
        Outfix(intty.r1,2,9);Outfix(intty.r2,2,9);Outimage;
        Outtext("GIVE NEW VALUES");Outimage;intty.r1:=Inreal;
        intty.r2:=Inreal;newpage;re_draw_edited;
    END ELSE
```

```
IF opcode=22  THEN
BEGIN
    Outtext("RDRAWTO");Outimage;Outimage;
    Outfix(intty.r1,2,9);Outfix(intty.r2,2,9);Outimage;
    Outtext("GIVE NEW VALUES");Outimage;
    intty.r1:=Inreal;intty.r2:=Inreal;
    newpage;re_draw_edited;
END ELSE


IF opcode=23  THEN
BEGIN
    IF  log_error THEN
    BEGIN
        cont_error:=TRUE;
        WHILE cont_error DO
        BEGIN
            IF (intty.r1 <= 0 ) THEN cont_error:=
            TRUE ELSE cont_error:=FALSE;
            IF cont_error THEN
            BEGIN
                Outtext("ERROR  IN DATA _MPOLAR ");Outimage;
                Outfix(intty.r1,2,9);Outfix(intty.r2,2,9);
                Outimage;
                Outtext("GIVE CORRECT VALUES");Outimage;
                intty.r1:=Inreal;intty.r2:=Inreal;

            END;
        END;
    END  ELSE  BEGIN

        Outtext("MPOLAR");Outimage;Outimage;
        Outfix(intty.r1,2,9);Outfix(intty.r2,2,9);Outimage;
        Outtext("GIVE NEW VALUES");Outimage;intty.r1:=Inreal;
        intty.r2:=Inreal;newpage;re_draw_edited;
    END;
END ELSE

IF opcode=24  THEN
BEGIN
    IF  log_error THEN
    BEGIN
        cont_error:=TRUE;
        WHILE cont_error DO
        BEGIN
            IF (intty.r1 <= 0 ) THEN
            cont_error:=TRUE ELSE cont_error:=FALSE;
            IF cont_error THEN BEGIN
```

```
                    Outtext("ERROR IN DATA  DPOLAR    ");Outimage;
                    Outfix(intty.r1,2,9);Outfix(intty.r2,2,9);Outimage;
                    Outtext("GIVE  CORRECT  VALUES");Outimage;
                    intty.r1:=Inreal;intty.r2:=Inreal;
                END;
            END;
        END ELSE  BEGIN
            Outtext("DPOLAR");Outimage;Outimage;
            Outfix(intty.r1,2,9);Outfix(intty.r2,2,9);Outimage;
            Outtext("GIVE NEW VALUES");Outimage;intty.r1:=Inreal;
            intty.r2:=Inreal;newpage;re_draw_edited;
        END;
END
ELSE
IF opcode=25  THEN
BEGIN
        IF log_error THEN
        BEGIN
            cont_error:=TRUE;
            WHILE cont_error DO
            BEGIN
                IF intty.r1 <= 0 THEN cont_error:=
                TRUE ELSE cont_error:=FALSE;
                IF cont_error  THEN
                BEGIN
                    Outtext("ERROR IN DATA ARC");Outimage;
                    Outfix(intty.r1,2,9);Outfix(intty.r2,2,9);
                    Outfix(intty.r3,2,9);Outimage;
                    Outtext("GIVE CORRECT VALUES");Outimage;
                    intty.r1:=Inreal;
                    intty.r2:=Inreal;intty.r3:=Inreal;
                END;
            END;
        END
        ELSE
        BEGIN
            Outtext("ARC");Outimage;Outimage;
            Outfix(intty.r1,2,9);Outfix(intty.r2,2,9);
            Outfix(intty.r3,2,9);Outimage;

            Outtext("GIVE NEW VALUES");Outimage;intty.r1:=Inreal;
            intty.r2:=Inreal;intty.r3:=Inreal;
            newpage;re_draw_edited;
        END;
END
ELSE
IF opcode=36  THEN
BEGIN
        IF log_error THEN
        BEGIN
            cont_error:=TRUE;
```

```
        WHILE cont_error DO
        BEGIN
            st_line;
            IF stline THEN cont_error :=TRUE ELSE cont_error:=FALSE;
            IF cont_error THEN
            BEGIN
                Outtext("ERROR IN DATA ARC3PT");Outimage;

                Outtext("GIVE CORRECT VALUES");intty.r1:=Inreal;
                intty.r2:=Inreal;
                intty.r3:=Inreal;intty.r4:=Inreal;
                intty.r5:=Inreal;intty.r6:=Inreal;

            ;END;
        END;
    END ELSE
    BEGIN


        Outtext("ARC3PT");Outimage;Outimage;
        Outfix(intty.r1,2,9);Outfix(intty.r2,2,9);
        Outfix(intty.r3,2,9);
        Outfix(intty.r4,2,9);Outfix(intty.r5,2,9);
        Outfix(intty.r6,2,9);
        Outimage;
        Outtext("GIVE NEW VALUES");Outimage;intty.r1:=Inreal;
        intty.r2:=Inreal;intty.r3:=Inreal;intty.r4:=Inreal;
        intty.r5:=Inreal;intty.r6:=Inreal;
        newpage;re_draw_edited;
    END ;
END
ELSE
IF opcode=34 THEN
BEGIN

    IF log_error THEN
    BEGIN
        cont_error:=TRUE;
        WHILE cont_error DO
        BEGIN
            IF (intty.r3 <= 0 OR intty.i1 <= 0
            OR intty.i2 <= 0)  THEN cont_error:=TRUE
            ELSE cont_error:=FALSE;
            IF cont_error THEN
            BEGIN
                Outtext("DATA ERROR  CIRPANEL");Outimage;
                Outfix(intty.r1,2,9);Outfix(intty.r2,2,9);
                Outfix(intty.r3,2,9);
                Outint(intty.i1,3);Outint(intty.i2,3);
                Outimage;
                Outtext("GIVE CORRECT VALUES");Outimage;
```

```
                    intty.r1:=Inreal;intty.r2:=Inreal;intty.r3:=Inreal;
                    intty.i1:=Inint;intty.i2:=Inint;
                END;
            END;
        END
        ELSE BEGIN
            Outtext("CIRPANEL");Outimage;Outimage;
            Outfix(intty.r1,2,9);Outfix(intty.r2,2,9);
            Outfix(intty.r3,2,9);
            Outint(intty.i1,3);Outint(intty.i2,3);Outimage;
            Outtext("GIVE NEW VALUES");Outimage;intty.r1:=Inreal;
            intty.r2:=Inreal;intty.r3:=Inreal;intty.i1:=Inint;
            intty.i2:=Inint;newpage;re_draw_edited;
        END;
    END ELSE

    IF opcode=35 THEN
    BEGIN
        IF log_error THEN
        BEGIN
            cont_error:=TRUE;
            WHILE cont_error DO
            BEGIN
                IF (intty.i1 <= 0 OR intty.i2 <= 0
                OR intty.i3 <= 0) THEN cont_error:=TRUE
                ELSE cont_error:=FALSE;
                IF cont_error THEN BEGIN
                    Outtext("ERROR  IN DATAPOLYPANEL");Outimage;
                    Outtext("GIVE  CORRECT  VALUES");intty.i1:=Inint;
                END;
            END;
        END  ELSE
        BEGIN
            Outtext("POLYPANEL");Outimage;Outimage;
            Outtext("GIVE NEW VALUES");Outimage;intty.r1:=Inreal;
            intty.r2:=Inreal;intty.r3:=Inreal;intty.i1:=Inint;
            intty.r5:=Inreal;intty.r6:=Inreal;
            newpage;re_draw_edited;
        END;
    END ELSE


    IF opcode= 37 THEN
    BEGIN
        Outtext("ARCPANEL");Outimage;Outimage;
        Outfix(intty.r1,2,9);Outint(intty.i1,3);Outint(intty.i2,8);
        Outint(intty.i3,3);Outint(intty.i4,3);Outimage;
        !Outtext("GIVE NEW VALUES");Outimage;intty.r1:=Inreal;
        intty.i1:=Inint;intty.i2:=Inint;intty.i3:=Inint;
    intty.i4:=Inint;newpage;re_draw_edited;END ELSE
    IF opcode=38 THEN  BEGIN
```

```
    Outtext("CIR3PT");Outimage;Outimage;
    Outfix(intty.r1,2,9);Outfix(intty.r2,2,9);Outfix(intty.r3,2,9);
    Outfix(intty.r4,2,9);Outfix(intty.r5,2,9);Outfix(intty.r6,2,9);
    Outimage;
    Outtext("GIVE NEW VALUES");Outimage;intty.r1:=Inreal;
    intty.r2:=Inreal;intty.r3:=Inreal;intty.r4:=Inreal;
    intty.r5:=Inreal;intty.r6:=Inreal;
newpage;re_draw_edited;END ELSE IF opcode=39 THEN  BEGIN
    Outtext("CHDRAW");Outimage;Outimage;
    Outtext("GIVE NEW VALUES");Outimage;intty.r1:=Inreal;
    intty.r2:=Inreal;intty.i1:=Inint;
newpage;re_draw_edited;END ELSE


IF opcode=40 THEN  BEGIN
    IF log_error THEN  BEGIN
        cont_error:=TRUE;
        WHILE cont_error DO BEGIN
            IF (intty.r3 <= 0 OR intty.i1 <= 0)
            THEN cont_error:=TRUE ELSE
            cont_error:=FALSE;
            IF cont_error THEN BEGIN
                Outtext("ERROR IN DATAREGPOL");
                Outimage;Outtext("GIVE  CORRECT VALUES");
                Outimage;intty.r1:=Inreal;
                intty.r2:=Inreal;intty.r3:=Inreal;
                intty.i1:=Inint;
            ;END;
        END;
    END
    ELSE BEGIN
        Outtext("REGPOL");Outimage;Outimage;
        Outtext("GIVE NEW VALUES");Outimage;intty.r1:=Inreal;
        intty.r2:=Inreal;intty.r3:=Inreal;intty.i1:=Inint;
        newpage;re_draw_edited;
    END
;END ELSE


IF opcode=26  THEN
BEGIN
    Outtext( "    SHEAR_X ");Outimage;
    Outtext("GIVE NEW VALUE");Outimage;intty.r1:=Inreal;
    ed_temp(1,1):=ed_temp(2,2):=ed_temp(3,3):=1.0;
    ed_temp(2,1):=intty.r1;ed_temp(1,2):=
    ed_temp(1,3):=ed_temp(2,3):=
    ed_temp(3,1):=ed_temp(3,2):=0;ed_trfmul;newpage;
    re_draw_edited;
END  ELSE

IF opcode=29  THEN
```

```
BEGIN
     Outtext( "    SCALE ");Outimage;
     Outfix(intty.r1,2,9);Outfix(intty.r2,2,9);Outimage;
     Outtext("GIVE NEW VALUES");Outimage;intty.r1:=Inreal;
     intty.r2:=Inreal;
     ed_temp(1,1):=intty.r1;ed_temp(2,2):=intty.r2;
     ed_temp(3,3):=1.0;ed_temp(1,2):=ed_temp(1,3):=ed_temp(2,1):=
     ed_temp(2,3):=ed_temp(3,1):=ed_temp(3,2):=0;
     ed_trfmul;
     newpage;re_draw_edited;
END  ELSE
IF opcode=28  THEN  BEGIN
     Outtext( "    SCALE_UNI ");Outimage;
     Outfix(intty.r1,2,9);Outimage;
     Outtext("GIVE NEW VALUE");Outimage;intty.r1:=Inreal;
     ed_temp(1,1):=ed_temp(2,2):=intty.r1;ed_temp(1,2):=
     ed_temp(1,3):=
     ed_temp(2,1):=ed_temp(2,3):=ed_temp(3,1):=ed_temp(3,2):=0;
     ed_temp(3,3):=1.0;
     ed_trfmul;
     newpage;re_draw_edited;
END  ELSE
IF opcode=27  THEN  BEGIN
     Outtext( "    SHEAR_Y ");Outimage;
     Outtext("GIVE NEW VALUE");Outimage;Outfix(intty.r1,2,9);
     Outimage;intty.r1:=Inreal;

     ed_temp(1,2):=intty.r1;ed_temp(2,2):=ed_temp
     (1,1):=ed_temp(3,3):=1.0;
     ed_temp(1,3):=ed_temp(2,1):=ed_temp(2,3):=
     ed_temp(3,1):=ed_temp(3,2):=0;
     ed_trfmul;
     newpage;re_draw_edited;
END
ELSE
IF opcode=30  THEN
BEGIN
     Outtext( " SCALE_PT  ");Outimage;
     Outtext("GIVE NEW VALUES");Outimage;
     intty.r1:=Inreal;
     intty.r2:=Inreal;intty.r3:=Inreal;
     intty.r4:=Inreal;
     newpage;re_draw_edited;
END  ELSE
IF opcode=39  THEN  BEGIN
     Outtext( "CHDRAWTO     ");Outimage;
     Outtext("GIVE NEW VALUES");Outimage;
     intty.r1:=Inreal;
     intty.r2:=Inreal;intty.i1:=Inint;
     newpage;re_draw_edited;
END  ELSE
```

```
            IF opcode=41  THEN  BEGIN
                IF log_error THEN BEGIN
                    cont_error:=TRUE;
                    WHILE cont_error DO BEGIN
                        IF (intty.r3 <= 0 OR intty.i1 <= 0
                        OR intty.i2 <= 0 OR intty.i3 <= 0 )
                        THEN cont_error:=TRUE
                        ELSE cont_error:=FALSE;
                        IF cont_error THEN BEGIN
                            Outtext(
                            "DATA ERROR REGPOLPANEL");Outimage;
                            Outtext("GIVE CORRECT VALUES");
                            Outimage;
                            intty.r1:=Inreal;intty.r2
                            :=Inreal;intty.r3:=Inreal;
                            intty.i1:=Inint;intty.i2:=
                            Inint;intty.i3:=Inint;
                            ;
                        END;
                    END;
                END ELSE
                BEGIN
                    Outtext( "REGPOLPANEL  ");Outimage;
                    Outtext("GIVE NEW VALUES");
                    Outimage;intty.r1:=Inreal;
                    intty.r2:=Inreal;intty.r3:=
                    Inreal;intty.i1:=Inint;
                    intty.i2:=Inint;intty.i3:=Inint;
                    newpage;re_draw_edited;
                END;
            END;
        END    ELSE
        BEGIN

            ed_status.satisfied:=TRUE;
            ed_status.count:=0;
        END;
        ed_status.count:=ed_status.count+1;
    END
    ELSE ed_status.count:=2;

END
ELSE ed_status.count:=2;
END;
PROCEDURE ed_iden;
BEGIN
    INTEGER i,j;
    FOR i:=1 STEP 1 UNTIL 3 DO
    FOR j:=1 STEP 1 UNTIL 3 DO
    ed_matform(i,j):=0.0;
    ed_matform(i,1):=ed_matform(2,2)
```

```
    :=ed_matform(3,3):=1.0;

END;

PROCEDURE ed_trfmul;
BEGIN
    REAL ARRAY ed_tpy(1:3,1:3);INTEGER i,j,k;
    FOR i:=1 STEP 1 UNTIL 3 DO
    FOR k:=1 STEP 1 UNTIL 3 DO
    BEGIN
        ed_tpy(i,k):=0.0;
        FOR j:=1 STEP 1 UNTIL 3 DO
        ed_tpy(i,k):=ed_tpy(i,k)+ed_matform(i,j)*ed_temp(j,k);
    END;
    FOR  i:=1 STEP 1 UNTIL   3 DO
    FOR j:=1 STEP 1 UNTIL 3 DO
    ed_matform(i,j):=ed_tpy(i,j);
END;

PROCEDURE re_draw_edited;
BEGIN
    INTEGER i;REAL x,y;
    re_drawing:=TRUE;
    IF c_ed_prgmcounter  > 0 THEN
    BEGIN
        FOR i:=1 STEP 1 UNTIL c_ed_prgmcounter DO
        BEGIN
            x:=temp_ed_attr(i,1)*ed_matform(1,1)+
            temp_ed_attr(i,2)*ed_matform(2,1)+
            ed_matform(3,1);
            y:=temp_ed_attr(i,1)*ed_matform(1,2)+
            temp_ed_attr(i,2)*ed_matform(2,2)+
            ed_matform(3,2);
            temp_ed_attr(i,1):=x;
            temp_ed_attr(i,2):=y;
            IF temp_ed_no_opcode(i,2)=9 THEN
            ed_moveto(x,y) ELSE
            ed_drawto(x,y);
        END;
    END;
    re_drawing:=FALSE;
END;

PROCEDURE bring_digit(i);INTEGER i;
BEGIN
    IF i=0 THEN
    BEGIN
        rmoveto(0,20);rdrawto(0,60);rdrawto(20,20);
        rdrawto(30,0);rdrawto(20,-20);
        rdrawto(0,-60);rdrawto(-20,-20);
```

```
    rdrawto(-30,0);rdrawto(-20,20);
    rmoveto(20,-20);rmoveto(-20,20);
    rdrawto(70,60);rmoveto(20,20);
END
ELSE
IF i=1 THEN
BEGIN
    rmoveto(20,80);rdrawto(15,20);rdrawto(0
    ,-100);rmoveto(-17.5,0);
    rdrawto(35,0);
    rmoveto(40,100);
END
ELSE
IF i=2 THEN
BEGIN
    rmoveto(0,70);rdrawto(0,10);rdrawto(
    20,20);rdrawto(30,0);rdrawto(20,-20);
    rdrawto(0,-20);rdrawto(-70,-40);
    rdrawto(0,-20);rdrawto(70,0);
    rmoveto(20,100);
END
ELSE
IF i=3 THEN
BEGIN
    rmoveto(0,80);rmoveto(20,20);rdrawto
    (30,0);rdrawto(20,-20);
    rdrawto(0,-10);rdrawto(-20,-20);rdrawto(-20,0);
    rmoveto(20,0);rdrawto(20,-20);rdrawto(
    0,-10);rdrawto(-20,-20);
    rdrawto(-30,0);
    rmoveto(90,100);
END
ELSE
IF i=4 THEN
BEGIN
    rmoveto(40,100);rdrawto(-40,-70);
    rdrawto(70,0);rmoveto(-30,70);
    rdrawto(0,-100);rmoveto(50,100);
END
ELSE
IF i=5 THEN
BEGIN
    rmoveto(70,100);rdrawto(-50,0);rdrawto(-10,-10);
    rdrawto(0,-30);rdrawto(10,-10);rdrawto(40,0);
    rdrawto(10,-10);rdrawto(0,-30);
    rdrawto(-10,-10);rdrawto(-40,0);
    rdrawto(-10,10);rmoveto(80,90);
END
ELSE
IF i=6 THEN
BEGIN
```

```
        rmoveto(60,80);rdrawto(-20,20);
        rdrawto(-20,0);rdrawto(-20,-20);
        rdrawto(0,-60);rdrawto(20,-20);
        rdrawto(20,0);rdrawto(20,20);
        rdrawto(0,20);rdrawto(-20,20);
        rdrawto(-20,0);rdrawto(-20,-20);
        rmoveto(90,60);
    END
    ELSE
    IF i=7 THEN
    BEGIN
        rmoveto(10,100);rdrawto(50,0);rdrawto(
        -40,-100);rmoveto(70,100);
    END
    ELSE
    IF i=8 THEN
    BEGIN
        rmoveto(20,100);rdrawto(-10,-10);
        rdrawto(0,-30);rdrawto(10,-10);
        rdrawto(30,0);rdrawto(10,-10);
        rdrawto(0,-30);rdrawto(-10,-10);
        rdrawto(-30,0);rdrawto(-10,10);rdrawto
        (0,30);rdrawto(10,10);rdrawto(30,0);
        rdrawto(10,10);rdrawto(0,30);
        rdrawto(-10,10);rdrawto(-30,00);
        rmoveto(-30,0);rmoveto(70,0);
    END
    ELSE
    IF i=9 THEN
    BEGIN
        rmoveto(70,60);rdrawto(-20,-20);rdrawto(-20,0);
        rdrawto(-20,20);rdrawto(0,20);
        rdrawto(20,20);rdrawto(20,0);
        rdrawto(20,-20);rdrawto(0,-60);
        rdrawto(-20,-20);rdrawto(-20,0);
        rdrawto(-20,20);rmoveto(80,80);
    END;
END;
PROCEDURE bring_char(i);CHARACTER i;
BEGIN
    IF i='A' THEN
    BEGIN
        rdrawto(35,100);rdrawto(35,-100);
        rmoveto(-52.5,50);rdrawto(35,0);
        rmoveto(37.5,50);
    END
    ELSE
    IF i='B' THEN
    BEGIN
        rmoveto(0,100);rdrawto(60,0);
        rdrawto(10,-10);rdrawto(0,-30);
```

```
    rdrawto(-20,-10);
    rdrawto(20,-10);rdrawto(0,-30);
    rdrawto(-10,-10);rdrawto(-60,0);
    rmoveto(0,50);rdrawto(50,0);
    rmoveto(-50,0);rmoveto(0,50);rdrawto(0,-100);
    rmoveto(0,100);
    rmoveto(80,0);
END
ELSE
IF i='C' THEN
BEGIN
    rmoveto(70,80);rdrawto(-20,20);rdrawto(-30,0);
    rdrawto(-20,-20);rdrawto(0,-60);rdrawto(20,-20);
    rdrawto(30,0);rdrawto(20,20);
    rmoveto(20,80);
END
ELSE
IF i='D' THEN
BEGIN
    rmoveto(0,100);rdrawto(50,0);
    rdrawto(20,-20);rdrawto(0,-60);
    rdrawto(-20,-20);rdrawto(-50,0);rmoveto(10,100);rdrawto(0,-100);
    rmoveto(80,100);
END
ELSE
IF i='E' THEN
BEGIN
    rmoveto(70,100);rdrawto(-60,0);rdrawto(0,-100);
    rdrawto(60,0);rmoveto(-60,50);rdrawto(40,0);
    rmoveto(40,50);
END
ELSE    IF i='F' THEN
BEGIN
    rmoveto(70,100);rdrawto(-60,0);rdrawto(0,-100);
    rmoveto(0,50);rdrawto(40,0);rmoveto(40,50);
END
ELSE
IF i='G' THEN
BEGIN
    rmoveto(40,50);rdrawto(30,0);rdrawto(0,-30);rdrawto(-20,-20);
    rdrawto(-30,0);
    rdrawto(-20,20);rdrawto(0,60);rdrawto(20,20);
    rdrawto(30,0);rdrawto(20,-20);

    rmoveto(20,20);
END
ELSE
IF i='H' THEN
BEGIN
    rmoveto(10,100);rdrawto(0,-100);rmoveto(0,50);
    rdrawto(50,0);rmoveto(0,50);rdrawto(0,-100);
```

```
BEGIN
    rmoveto(10,0);rdrawto(0,100);rdrawto(50,0);rdrawto(10,-10);
    rdrawto(0,-30);rdrawto(-10,-10);
    rdrawto(-50,0);rmoveto(80,50);
END
ELSE

IF i='Q' THEN
BEGIN
    rmoveto(20,0);rdrawto(30,0);rdrawto(20,20);rdrawto(0,60);
    rdrawto(-20,20);rdrawto(-30,0);
    rdrawto(-20,-20);rdrawto(0,-60);rdrawto(20,-20);
    rmoveto(30,20);rdrawto(20,-20);rmoveto(20,100);
END
ELSE
IF i='R' THEN
BEGIN
    rmoveto(10,100);rdrawto(0,-100);
    rmoveto(-10,100);rdrawto(60,0);rdrawto(10,-10);
    rdrawto(0,-30);rdrawto(-10,-10);
    rdrawto(-50,0);rdrawto(60,-50);
    rmoveto(20,100);
END
ELSE
IF i='S' THEN
BEGIN
    rmoveto(70,90);rdrawto(-10,10);
    rdrawto(-50,0);rdrawto(-10,-10);
    rdrawto(0,-30);rdrawto(10,-10);
    rdrawto(50,0);rdrawto(10,-10);
    rdrawto(0,-30);rdrawto(-10,-10);
    rdrawto(-50,0);rdrawto(-10,10);
    rmoveto(90,90);
END
ELSE
IF i='T' THEN
BEGIN
    rmoveto(0,100);rdrawto(70,0);
    rmoveto(-35,0);rdrawto(0,-100);
    rmoveto(55,100);
END
ELSE
IF i='U' THEN
BEGIN
    rmoveto(0,100);rdrawto(0,-80);
    rdrawto(20,-20);rdrawto(30,0);rdrawto(20,20);rdrawto(0,80);
    rmoveto(20,0);
END
ELSE
IF i='V' THEN
BEGIN
```

```
        rmoveto(0,100);rdrawto(35,-100);
        rdrawto(35,100);rmoveto(20,0);
    END
    ELSE
    IF i='W' THEN
    BEGIN
        rmoveto(0,100);rdrawto(0,-100);
        rdrawto(35,50);rdrawto(35,-50);
        rdrawto(0,100);rmoveto(20,0);
    END
    ELSE
    IF i='X' THEN
    BEGIN
        rdrawto(70,100); rmoveto(-70,0);rdrawto(70,-100);
    END
    ELSE
    IF i='Y' THEN
    BEGIN
        rdrawto(70,100);rmoveto(-70,0);rdrawto(35,-50);
        rmoveto(55,50);
    END
    ELSE
    IF i='Z' THEN
    BEGIN
        rmoveto(0,100);rdrawto(70,0);rdrawto(-70,-100);rdrawto(70,0);
        rmoveto(20,100);
    END;
END;
PROCEDURE bringdigit(x,y,i);REAL x,y;INTEGER i;
BEGIN
    INTEGER k,j,l;
    IF NOT(Open AND edit ) THEN
    BEGIN
        complex:=TRUE;opcode:=102;k:=1;l:=5;
        save_wvconstants;
        WHILE k<5 DO
        BEGIN
            getback;moveto(x,y);setwindow(0,780/sca_d_x,0,780/sca_d_y);
            bring_digit(i);GO TO retdig;
            IF k<3 THEN j:=0 ELSE
            BEGIN
                j:=1;l:=0;
            END;
            IF Mod(k,2)=0 THEN
            BEGIN
                j:=-j;l:=-l;
            END;
            setviewport(chvxl+l,chvxh+l,chvyl+j,chvyt+j);
            k:=k+1;
            IF k=5 THEN
            BEGIN
```

```
                getback;moveto(x,y);setwindow(0,780/
                sca_d_x,0,780/sca_d_y);
                bring_digit(i);
            END;
        END;
        retdig:         getback;
    END;
    complex:=FALSE;
END;
PROCEDURE bringchar(x,y,i);REAL x,y;CHARACTER i;
BEGIN
    INTEGER l,j,k;
    IF NOT(open AND edit) THEN
    BEGIN
        complex:=TRUE;opcode:=101;l:=5;
        k:=1;save_wvconstants;
        WHILE k<5 DO
        BEGIN
            getback;moveto(x,y);setwindow(0,780/
            sca_ch_x,0,780/sca_ch_y);
            bring_char(i);GO TO retcha;
            IF k<3 THEN j:=0 ELSE
            BEGIN
                j:=l;l:=0;
            END;
            IF Mod(k,2) =0 THEN
            BEGIN
                j:=-j;l:=-l;
            END;
            setviewport(chvxl+l,chvxh+l,chvyl+j,chvyt+j);
            k:=k+1;
            IF k=5 THEN
            BEGIN
                getback;
                moveto(x,y);
                setwindow(0,780/sca_ch_x,0,780/sca_ch_y);
                bring_char(i);exit;
            END;
        END;

        retcha:         getback;
    END;
    complex:=FALSE;
END;
PROCEDURE bringtext(x,y,tx);
REAL x,y;TEXT tx;
BEGIN
    INTEGER i,j;CHARACTER c;TEXT t;
    IF NOT(open AND edit) THEN
    BEGIN
        opcode:=100;complex:=TRUE;t:=tx;j:=t.Length;
```

```
            t.Setpos(1);
            FOR i:=1 STEP 1 UNTIL l DO
            BEGIN
                c:=t.Getchar;bringchar(x,y,c);
                x:=x+(wxrc-wxlc)*(100/780)*(sca_ch_x);
            END;
        END;
        IF NOT master THEN      complex:=FALSE;
    END;
    PROCEDURE scale_digit(sx,sy);
    REAL sx,sy;
    BEGIN
        sca_d_x:=sx;sca_d_y:=sy;
    END;
    PROCEDURE scale_char(sx,sy);REAL sx,sy;
    BEGIN
        sca_ch_x:=sx;sca_ch_y:=sy;
    END;
    GO TO ex_done;
    done:WHILE TRUE DO BEGIN Outimage;Outtext(" TYPE CONTROL C ");Inimage;END;


    ex_done:
    d_to_r:=3.14145926/180;init_gsim;
    r_to_d:=180/3.1415926;
    tcr:-Copy("TYPE");
    Outtext(" >>GSIMULA IN EXECUTION;;");
    Outimage;Outimage;slow;
    retlst:    break;newpage;slow;
END;
```

```
BEGIN
    EXTERNAL PROCEDURE rubout;
    EXTERNAL CLASS gsimul;gsimul
    BEGIN
        PROCEDURE heading;
        BEGIN
            TEXT hd;save_wvconstants;setwindow(0,900,0,900);
            setviewport(0,780,0,780);
            scale_char(.5,.5);hd:-Copy("GSIMULA");
            bringtext(225,800,hd);
            getback;
        END;
        PROCEDURE t_hanks;
        BEGIN
            TEXT t;
            scale_char(1,1);t:-Copy("THANKS");
            setwindow(0,100,0,100);setviewport(0,780,0,780);
            square(0,0,100);
            bringtext(20,50,t);
        END;
        PROCEDURE wellcome;
        BEGIN
            TEXT t;INTEGER i,j;CHARACTER c;REAL x,y;
            setwindow(0,780,0,780);setviewport(0,780,0,780);
            square(0,0,780);heading;setviewport(0,780,0,780);
            t:-Copy("WELCOME");i:=t.Length;t.Setpos(1);
            FOR j:=1 STEP 1 UNTIL i DO
            BEGIN
                c:=t.Getchar;x:=265+(j-1)*80*2;
                IF j<=4 THEN
                y:=400+(4-j)*50 ELSE y:=400+(j-4)*50-100+(6-j)*50;
                IF j<5 THEN scale_char(1,1+.2*(j-1))
                ELSE scale_char(1,.2*(7-j)+1);
                bringchar(x,y+(j-1)*25,c);
            END;
        END;
        PROCEDURE chardigit;
        BEGIN
            TEXT wel,wle,lew,lwe;
            wel:-Copy("ABCDEFG");wle:-Copy("HIJKLMN");
            lew:-Copy("OPQRSTU");
            lwe:-Copy("VWXYZ");
            setwindow(0,780,0,780);setviewport(0,780,0,780);
            square(0,0,780);scale_char(1,1);
            bringtext(100,500,wel);scale_char(.8,.8);
            bringtext(100,350,wle);scale_char(.6,.6);
            bringtext(100,200,lew);
            scale_char(.4,.4);      bringtext(100,50,lwe);
            scale_char(1,1);square(0,0,780);
            scale_digit(1,1);break;
            bringdigit(100,500,1);bringdigit(200
```

```
    ,500,2);bringdigit(300,500,3);
    bringdigit(400,500,4);bringdigit(500,500,5);
    scale_digit(.8,.8);
    bringdigit(100,350,6);bringdigit
    (200,350,7);bringdigit(300,350,8);
    bringdigit(400,350,9);bringdigit(500,350,0);
END;
PROCEDURE logicerror;
BEGIN
    setwindow(0,780,0,780);setviewport(0,780,0,780);
    square(0,0,780);
    setwindow(100,-100,100,1000);rectangle(50,50,-90,9);
END;
PROCEDURE cirpattern;
BEGIN
    INTEGER  j, n,i,nm1,ip1;
    REAL r,thinc,theta;REAL ARRAY x(1:100),y(1:100);
    setwindow(-500,500,-500,500);setviewport
    (0,780,0,780);n:=30;r:=500;
    thinc:=6.281385307/n;theta:=0.0;
    square(-500,-500,1000);
    FOR i:=1 STEP 1 UNTIL n DO
    BEGIN
        theta:=theta+thinc;x(i):=r
        *Cos(theta);y(i):=r*Sin(theta);
    END;nm1:=n-1;
    FOR i:=1 STEP 1 UNTIL nm1 DO
    BEGIN
        ip1:=i+1;
        FOR j:=ip1 STEP 1 UNTIL n DO BEGIN
            moveto(x(i),y(i));drawto(x(j),y(j));
        END;
    END;
END;
PROCEDURE squarepat;
BEGIN
    REAL ARRAY x(1:4),y(1:4),xd(1:4),yd
    (1:4);REAL smu,rmu;INTEGER i,j,nj;
    IF  sqpat_left THEN  BEGIN
        x(1):=x(2):=500;x(3):=x(4):=-500;y(1):=y(4):=500;
        y(2):=y(3):=-500;
    END ELSE BEGIN y(1):=y(2):=500
        ;y(3):=y(4):=-500;x(1):=x(4):=500;
        x(2):=x(3):=-500;
    END;
    setwindow(-500,500,-500,500);smu:=0.1;
    square(-500,-500,1000);
    rmu:=1.0-smu;
    FOR i:=1 STEP 1 UNTIL 21 DO
    BEGIN
        moveto(x(4),y(4));FOR j:=1 STEP 1 UNTIL 4 DO
```

```
        BEGIN
            drawto(x(j),y(j));nj:=Mod(j,4)+1;
            xd(j):=smu*x(j)+rmu*x(nj);
            yd(j):=smu*y(j)+rmu*y(nj);
        END;
        FOR j:=1 STEP 1 UNTIL 4 DO BEGIN
        x(j):=xd(j);y(j):=yd(j);END;
    END;
END;

PROCEDURE  spiro(a,b,d,scale);INTEGER a,b,d;REAL scale;
BEGIN
    REAL rd,rab,theta,thetad,aoverb,phi,x,y;INTEGER n,no,i,icf;
    PROCEDURE iicf(a,b);INTEGER a,b;
    BEGIN
        INTEGER m,i,j;i:=a;j:=b;
        one: m:=Mod(i,j);IF (m=0) THEN GO
        TO two;i:=j;j:=m;GO TO one;
        two: icf:=j;
    END;
    setwindow(-1.0,20.5,-1.0,20.5);setviewport(0,780,0,780);rd:=d*scale;
    theta:=0.0;square(0,0,18);
    rab:=(a-b)*scale;
    thetad:=3.1415926535*.02;aoverb:=a/b;          iicf(a,b);
    n:=b/icf;no:=n*100;moveto(11,8.3);
    FOR i:=1 STEP 1 UNTIL no DO
    BEGIN

        theta:=theta+thetad;phi:=theta*aoverb;
        x:=rab*Cos(theta)+rd*Cos(phi)+8;
        y:=rab*Sin(theta)-rd*Sin(phi)+10;
        IF i=1 THEN moveto(x,y) ELSE drawto(x,y);
    END;
END;
PROCEDURE cotton_pin(x,y,r,n);REAL x,y,r;INTEGER n;
BEGIN
    REAL ARRAY ax(1:4*n),ay(1:4*n),axx(0:39),ayy(0:39);
    REAL x1,y1,r1,n1,del;
    INTEGER i, j,mo;
    x1:=x;y1:=y;r1:=r;n1:=n;del:=r1/n;
    setwindow(-500,500,-500,500);
    setviewport(0,780,0,780);
    square(x1,y1,r1);
    FOR j:=1 STEP 1 UNTIL 11 DO
    BEGIN
        ax(j):=x1+(j-1)*del;ay(j):=y1;
        ax(32-j):=ax(j);ay(32-j):=y1+r1;
        IF j < 10 THEN BEGIN
            ax(j+11):=x1+r1;ay(j+11):=y1+j*del;

            ax(40-(j-1)):=x1;
```

```
            ay(40-(j-1)):=ay(j+11);
        END;
    END;
    FOR j:=0  STEP 1 UNTIL 39 DO BEGIN
        axx(j):=ax(j+1);ayy(j):=ay(j+1);
    END;
    FOR i :=0 STEP 1 UNTIL 39 DO
    BEGIN
        FOR j:=0 STEP 1 UNTIL  39  DO
        BEGIN
            BEGIN
                IF (j-i) > 0 THEN mo:=j-i ELSE mo:=j-i+40;
                IF (mo=1) OR (mo=2) OR (mo=3) OR (mo=5) OR (mo=8) OR (mo=13)
                OR (mo=21) OR (mo=34) THEN
                BEGIN
                  moveto(axx(i ),ayy(i ));drawto(axx(j ),ayy(j ));
                END;
            END;
        END;
    END;
END;
PROCEDURE sqdesign;
BEGIN
    setwindow(-500,500,-500,500);sqpat_left:=FALSE;
    setviewport(0,195,585,780);squarepat;
    setviewport(195,390,390,585); squarepat;
    setviewport(390,585,195,390); squarepat;
    setviewport(585,780,0,195); squarepat;
    setviewport(390,585,585,780); squarepat;
    setviewport(585,780,390,585); squarepat;
    setviewport(0,195,195,390); squarepat;
    setviewport(195,390,0,195); squarepat;
    sqpat_left:=NOT( sqpat_left);
    setviewport(0,195,0,195);squarepat;
    setviewport(0,195,390,585);squarepat;
    setviewport(195,390,195,390);squarepat;
    setviewport(195,390,585,780);squarepat;
    setviewport(390,585,0,195);squarepat;
    setviewport(390,585,390,585);squarepat;

    setviewport(585,780,195,390);squarepat;
    setviewport(585,780,585,780);squarepat;


END;
PROCEDURE paneltest;
BEGIN
    REAL ARRAY   vx(1:10),vy(1:10);
    vx(1):=100;vx(2):=230;vx(3):=400;vx(4):=270;vx(5):=500;
    vx(6):=600;vx(7):=550;vx(8):=100;vx(9):=400;vx(10):=50;
    vy(1):=100;vy(2):=280;vy(3):=220;vy(4):=0;vy(5):=230;
```

```
    vy(6):=180;vy(7):=330;vy(8):=550;vy(9):=300;vy(10):=400;
    setwindow(0,700,0,700);setviewport(0,780,0,780);square(0,0,700);
    setviewport(390,780,390,780);
    poly(10,vx,vy);polypanel(10,vx,vy,40,45);
    setviewport(0,390,390,780);regpol(350,350,350,6);
    regpolpanel(350,350,350,6,80,135);
    setviewport(0,390,0,390);circle(350,350,350);
    cirpanel(350,350,350,20,90);
    setviewport(390,780,0,390);cirpanel(350,350,350,40,1);
END;

PROCEDURE transform_test;
BEGIN
    PROCEDURE figure;
    BEGIN
        rectangle(18,38,50,30);moveto(18,68);
        drawto(28,88);drawto(78,88);drawto(68,68);
        moveto(68,38);drawto(78,58); drawto(78,88);
    END;
    setwindow(0,120,0,120);setviewport(0,780,0,780);square(0,0,120);
    setviewport(390,780,390,780);figure;
    setviewport(0,390,390,780);shear_y(.5);figure;
    exit;setviewport(0,390,0,390);shear_x(.5);
    figure;exit;
    setviewport(390,780,0,390);scale_uni(.6);
    figure;exit;break;newpage;
    setwindow(-100,100,-100,100);setviewport(0,780,0,780);
    square(-100,-100,200);
    figure;
    reflect_x;
    figure;exit;
    reflect_y;figure;exit;
    reflect_xoy;figure;exit;
END;
PROCEDURE regpolpanel_test;
BEGIN
    setwindow(0,100,0,100);setviewport(390,780,390,780);
    regpol(50,50,50,6);break;regpolpanel(50,50,50,6,20,30);
    setviewport(0,390,0,390);regpol(50,50,50,9);
    regpolpanel(50,50,50,9,40,120);setviewport(0,390,0,390);
    regpol(50,50,50,30);
END;

PROCEDURE relative_test;
BEGIN
    REAL i,j;
    setwindow(0,100,0,100);setviewport(0,780,0,780);
    square(0,0,100);
    moveto(0,0);drawto(20,10);
    WHILE (( i < 1000) OR (i < 1000)) DO
```

```
    BEGIN
        i:=Inreal;j:=Inreal;
        rdrawto(i,j);
        Outimage;Inimage;
    END;
END;
PROCEDURE graph_plot_test;
BEGIN
    INTEGER  ARRAY vx(1:10),vy(1:10);TEXT s1,s2;
    s1:-Copy("X - AXIS");s2:-Copy("Y-AXIS");
    setwindow(0,100,0,100);setviewport(0,780,0,780);
    vx(1):=vy(1):=0;
    vx(2):=15;vy(2):=30;vx(3):=vy(3):=25;
    vx(4):=35;vy(4):=42;vx(5):=48;vy(4):=47;
    vx(6):=56;vy(6):=62;vx(7):=70;vy(7):=52;
    vx(8):=vy(8):=80;vx(9):=95;vy(9):=88;
    vx(10):=vy(10):=10;graphplot(10,vx,vy,s1,s2);
END;
PROCEDURE transformtest;
BEGIN
    break;
    setwindow(0,100,0,100);setviewport(0,780,0,780);
    square(0,0,100);exit;
    square(50,50,25);moveto(12,34);rotatept(30,75,50);
    square(50,50,25);
    exit;
    moveto(89,90);
    scalept(1,1.2,50,50);rotatept(-30,50,50);square(50,50,25);exit;
END;
PROCEDURE segmenting_test;
BEGIN
    REAL p,q;INTEGER i;REAL ARRAY ax(1:4),ay(1:4);
    REAL k;
    setwindow(-100,100,-100,100);
    open_segment(1);square(p,p,20);
    IF (sgpat_left) THEN BEGIN ax(1):=ax(4):=ay(1):=
        ay(2):=p;ax(2):=ax(3):=ay(3):=ay(4):=p+12;
    polypanel(4,ax,ay,10,1);END;
    close_segment(1);
    p:=q:=0;
    post(1);reflect_x;post(1);reflect_y;
    post(1);reflect_xoy;post(1);p:=q:=20;
    FOR i:=1 STEP 1 UNTIL 16 DO BEGIN
        IF i=1 THEN k:=0 ELSE k:=1;
        scale(1-.1*i,1-.1*i);translate(p,p);post(1);
        scale(1-.1*i,1-.1*i);translate(-p-18,-p-18);post(1);
        scale(1-.1*i,1-.1*i);translate(-p-18,p);post(1);
        scale(1-1*.1,1-.1*i);translate(p,-p-18);post(1);
    q:=q*0.9;p:=p+q;END;
END;
PROCEDURE seg_test;
```

```
BEGIN
    sqpat_left:=TRUE;
    setwindow(-100,100,-100,100);setviewport(0,780,0,780);
    square(-100,-100,200);
    segmenting_test;GO TO retseg;
    setwindow(-100,100,-100,100);setviewport(0,780,260,520);
    square(-100,-100,200);
    setviewport(0,390,260,520);segmenting_test;
    setviewport(390,780,260,520);segmenting_test;
    retseg:
END;


PROCEDURE arcpaneltest;
BEGIN
    REAL q;
    setwindow(0,100,0,100);setviewport(0,780,0,780);
    q:=60*d_to_r;square(0,0,100);
    moveto(100,50);arcpanel(50,0,60,30,1);
    moveto(50+50*Cos(q),50+50*Sin(q));
    arcpanel(50,60,120,15,91);
    moveto(50-50*Cos(q),50+50*Sin(q));
    arcpanel(50,120,180,30,45);
    moveto(0,50);arcpanel(50,180,240,60,135);
    moveto(50-50*Cos(q),50-50*Sin(q));
    arcpanel(50,240,300,15,30);
    moveto(50+50*Cos(q),50-50*Sin(q));
    arcpanel(50,300,360,30,92);
END;
PROCEDURE chdrawto_test;
BEGIN
    INTEGER i;
    setwindow(0,500,0,500);setviewport(0,780,0,780);
    scale_digit(.3,.3);
    square(0,0,500);FOR i:=0 STEP 1 UNTIL 6 DO
    BEGIN
        bringdigit(80,(9-i)*50,i);
        moveto(110,(9-i)*50);chdrawto(550,(9-i)*50,1);
    END;
END;
PROCEDURE demo;BEGIN
    scale_char(1,1);scale_digit(.3,.3);
    wellcome;break;newpage;;
    logicerror;break;newpage;
    transformtest;break;newpage;
    paneltest;break;newpage;
    arcpaneltest;newpage;
    relative_test;break;newpage;
    transform_test;break;newpage;
    chardigit;break;newpage;
    chdrawto_test;break;newpage;
```

```
            seq_test;break;newpage;
            sqdesign;break;newpage;
            cotton_pin(-500,-500,1000,10);;break;newpage;
            cirpattern;break;newpage;
            spiro(12,8,6,.5);break;newpage;
            t_hanks;
        END;
        demo;
    END;
END
```